AD-A032 377 NAVAL POSTGRADUATE SCHOOL MONTEREY CALIF F/G 9/2
A MICROPROGRAMMABLE DATA ACQUISITION AND CONTROL SYSTEM (MIDAS --ETC(U)
SEP 76 J K PLUNKETT

UNCLASSIFIED NL

1 OF 2
AD A032377

BLANK PAGE

(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A Microprogrammable Data Acquisition and Control System (MIDAS IIA) with Application to Mean Meteorological Data

by

John Russell Plunkett

September 1976

Thesis Advisor: T. M. Houlihan

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)

A Microprogrammable Data Acquisition and Control System (MIDAS IIA) with Application to Mean Meteorological Data

5. TYPE OF REPORT & PERIOD COVERED

Master's Thesis; September 1976

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)

John Russell Plunkett

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Naval Postgraduate School
Monterey, California 93940

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

Naval Postgraduate School
Monterey, California 93940

12. REPORT DATE

September 1976

13. NUMBER OF PAGES

144

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

143p.

15. SECURITY CLASS. (of this report)

Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Data Acquisition System
Microprocessor
Microcomputer
Digital System

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The construction and operation of a fully automated microprogrammable data acquisition and control system (MIDAS IIA) with application to the sampling and mean averaging of meteorological data is reported. MIDAS IIA is designed to automatically collect periodic samples of various meteorological data in digital and analog forms, compute mean averages over selectable time intervals, and produce a permanent output record of the time

next page

DD FORM 1 JAN 73 1473 (Page 1)

EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

conti
averaged data.

The system consists of a microprocessor based on the Intel Corp. 8008 CPU, a 16-channel multiplexed analog-to-digital converter, a digital clock, an incremental digital cassette tape recorder, numerous meteorological data sensors, and a teletype for input/output. Details of system operation and programming are described.

A Microprogrammable Data Acquisition and Control System (MIDAS IIA) with Application to Mean Meteorological Data

by

John Russell Plunkett
Lieutenant-Commander, United States Navy
B.S., California State University at Long Beach, 1965

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL

September 1976

Author: John R. Plunkett

Approved by: Thomas Houlihan
Thesis Advisor

Edward W. Ford
Chairman, Department of Aeronautics

Jack R. Borsting
Academic Dean

ABSTRACT

The construction and operation of a fully automated microprogrammable data acquisition and control system (MIDAS IIA) with application to the sampling and mean averaging of meteorological data is reported. MIDAS IIA is designed to automatically collect periodic samples of various meteorological data in digital and analog forms, compute mean averages over selectable time intervals, and produce a permanent output record of the time averaged data.

The system consists of a microprocessor based on the Intel Corp. 8008 CPU, a 16-channel multiplexed analog-to-digital converter, a digital clock, an incremental digital cassette tape recorder, numerous meterological data sensors, and a teletype for input/output. Details of system operation and programming are described.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

A large share of the credit for MIDAS IIA belongs to Lt. James W. Sturges who conceived the design and initiated its development. He continued to provide advice and consultation throughout the course of system development and his assistence is deeply appreciated.

Dr. Thomas M. Houlihan, the author's thesis advisor, provided invaluable support and encouragement in the development, construction, and documentation of the MIDAS IIA system.

The timely completion of MIDAS IIA was due in large part to the technical knowledge and skill of Mr. Ray Garcia, who assembled and helped interface the system hardware. The invaluable technical assistence of Mr. Tom Christian is likewise acknowledged.

Finally, much of the credit for MIDAS IIA goes indirectly to the author's wife, who has lived for the last year with a part-time husband. Her understanding and support were significant factors in its successful completion.

I. INTRODUCTION

The system described in this report was designed and built in response to the need for an automated data acquisition and control system to regulate the collection of periodic samples of meteorological data, compute mean averages over selectable intervals, and produce a permanent output record of the time averaged data.

The major design of this system is attributed to Lt. J. W. Sturges who had previously applied microprocessors to the control of data acquisition and recording (ref. 1). The present author completed the data system hardware, interfaced the various hardware components, programmed the system software in the PL/M language, and developed data reduction programs and procedures.

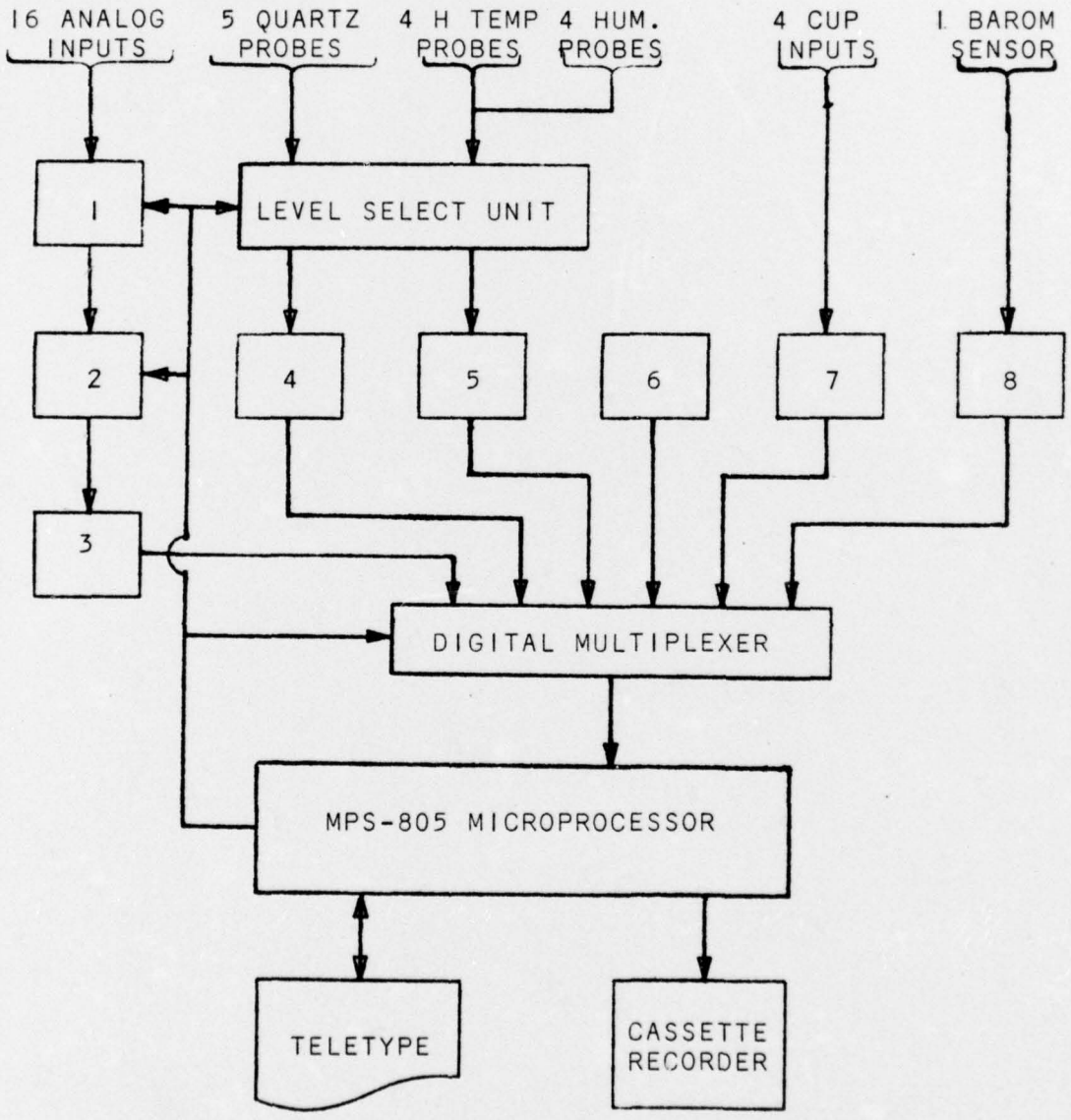
This document is intended to serve both as a system design report and as an operation manual. The main body of the report gives descriptions and general knowledge concerning the complete data acquisition and control system. Detailed descriptions of the software and hardware, along with procedures to modify the program, are contained in the appendices.

II. SYSTEM DESCRIPTION

A. GENERAL

MIDAS IIA is a microprocessor-based data acquisition system designed to control the sampling, averaging, and recording of atmospheric meteorological data. An overall functional block diagram of MIDAS IIA is shown in Fig 1, with the physical appearance of the system shown in Fig 2. Input data sources to MIDAS IIA include barometric pressure, temperature (two independent sources), relative humidity, cup anemometer wind velocity, and up to 16 analog signal inputs. At present, analog channels 1-4 are assigned to hot wire anemometer inputs.

Atmospheric data are sampled at four ascending heights above the sea surface with the object of determining an atmospheric profile. To accomplish this, sensor sets for temperature, relative humidity, and wind velocity are located on four mast platforms positioned at various heights above the deck of the research vessel R. V. ACANIA. A temperature sensor is also streamed overboard to obtain sea surface water temperature. Additionally, surface barometric pressure is sensed.



KEY:

1 - ANALOG MULTIPLEXER
2 - SAMPLE AND HOLD
3 - ANALOG-TO-DIGITAL CONV.
4 - QUARTZ THERMOMETER
5 - DIGITAL II INSTRUMENT
6 - DIGITAL CLOCK
7 - ANEMOMETER COUNTERS
8 - DIGITAL BAROMETER

Figure 1 - MIDAS IIA SYSTEM BLOCK DIAGRAM

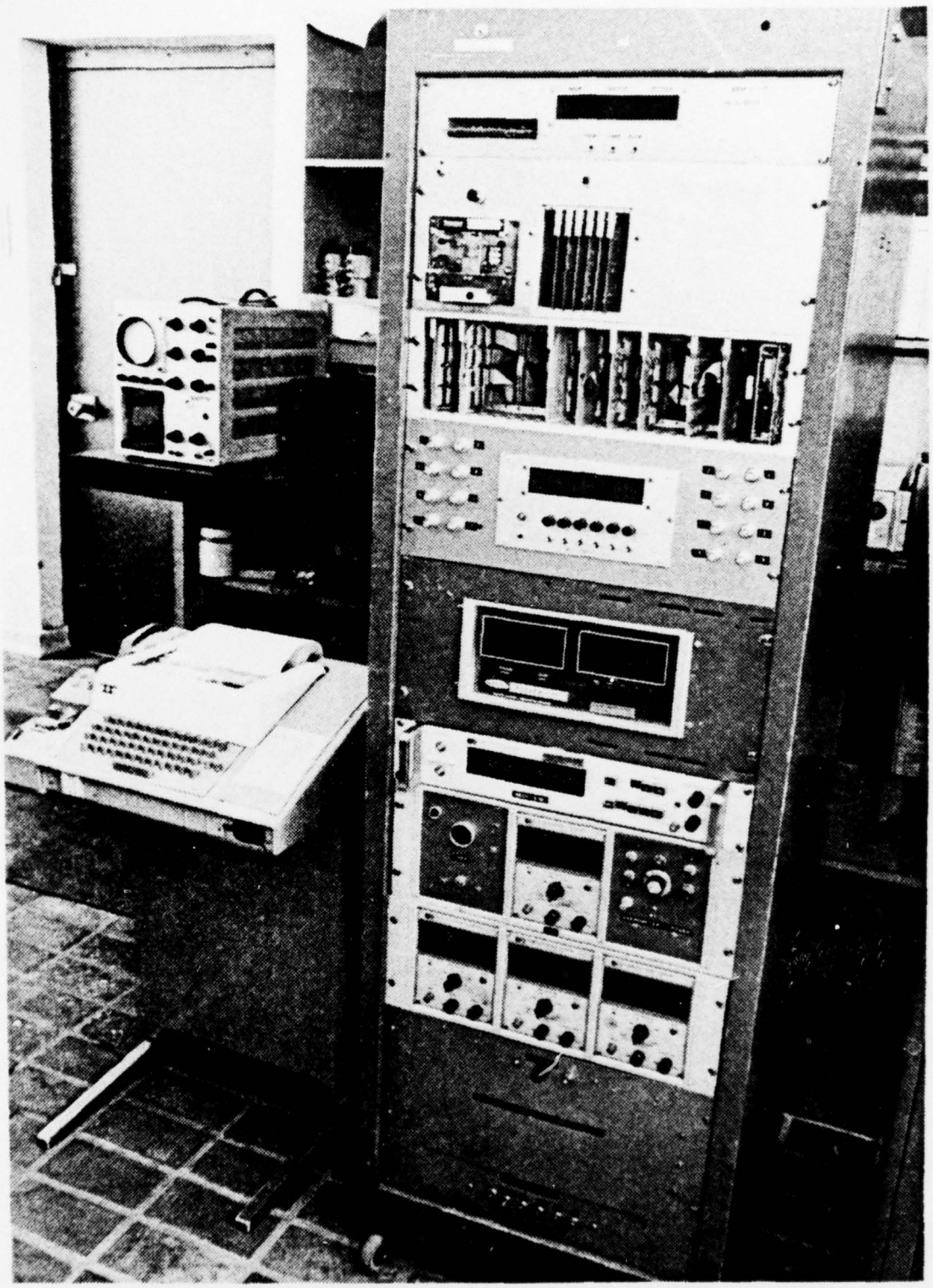


Figure 2 - MIDAS IIA EQUIPMENT CABINET AND TELETYPE

The MIDAS IIA system utilizes a microprocessor based on the Intel Corp. 8008 Central Processor Unit (CPU). All software programming is written in PL/M, an Intel "high level" language designed to support its microprocessors. The major advantage of using PL/M is that it facilitates the writing of a self-documenting program. This greatly simplifies the task of making future changes or additions.

The operator is interfaced with the system via Teletype for full duplex Input/Output communications and program control. The operator may alter the automatic sample list of sensors by deleting those which are not connected or are determined to be invalid. Also the operator exercises control over the sample start time and the number of samples to be averaged together before outputting. Once initiated, the system is fully automated to sample the tailored list of sensors every 30 seconds and periodically output values averaged over operator selected intervals.

Output values are printed out on the teletype in columnized format with the time of print as a header. The teletype has a paper tape punch incorporated which is normally activated by the operator to produce a data copy concurrent with the printout. Additionally, a magnetic cassette tape recorder automatically records the same output data, thus providing a redundancy factor to insure that a permanent output record is made.

A summary of the overall operational specifications of MIDAS IIA are shown in Table I, while the technical specifications are outlined in Table II.

Table I. MIDAS IIA Operational Specifications

1. Once initiated, the system will operate fully automatically to continually sample sensors every 30 seconds, compute averages at selected intervals, and output the mean (averaged) values.

2. Cassette tape recording is fully automatic once a tape is initiated.

3. All 16 analog input channels are active and available for user assignment as desired. Channels 1-4 are currently assigned to hot-wire anemometer inputs.

4. The current level selected is visually displayed on the Level Select Unit.

5. System start time is operator selectable in one minute increments. Start time is referenced to the system's 24 hour digital clock.

6. The averaging interval is operator selectable in one minute increments from 1 to 99.

7. The pre-set scan list can be modified by the operator to delete invalid sensors.

8. The Level Select Unit (Fig 4) multiplexes several sensor probes to the Quartz Thermometer instrument and the Digital II instrument. This feature of MIDAS IIA allows a single precision instrument to be used in the collection of high quality data from several different locations at minimum cost.

Table II. MIDAS IIA Technical Specifications

1. Sensor inputs:

   5 - Quartz Temperature
   4 - Hygrodynamics Temperature
   4 - Relative Humidity
   4 - Cup Anemometer
   1 - Barometer
  16 - Analog Signals

2. Sensor ranges:

   Quartz Temp. - 00.00° thru 49.99° Celsius.
   Hygrodynamics Temp. - 00.0° thru 99.9° Celsius or Farenheit.
   Relative Humidity - 00.0% thru 99.9%.
   Cup anemometer - 0 thru 255 revolutions in a 30 second time period.
   Barometer - 25.00 thru 31.00 inches Mercury
   Analog Signals - 0.00 thru 4.99 volts.

3. External power requirements: 115 VAC at 60 Hz.

4. Dual slope integration technique used in analog-to-digital conversion.

5. A discrete digital multiplexing scheme allows up to 32 eight-bit input channels (16 are presently implemented) to be connected to a single microprocessor input port.

B. MAJOR HARDWARE COMPONENTS

1. MPS-805 Microprocessor

A Pro-Log Corporation Model MPS-805 microprocessor, shown in Fig 3, is the heart of the MIDAS IIA system. Briefly, the MPS-805 is an 8-bit microcomputer built around the Intel Corporation 8008 Central Processor Unit (CPU). Reference 2 contains a complete description of the microprocessor hardware and its associated machine-level programming requirements.

Selection of the Pro-Log Corp. microprocessor was based on several factors. First of all the control program for MIDAS IIA was anticipated to be fairly large for a microcomputer, thus it was desired to use a "high-level" programming language. Since Intel Corp. was the pioneer in the microcomputer software support field with it's PL/1-based high level language, called PL/M, it was the logical choice. PL/M was designed to support Intel's CPU product line, namely the 8008 CPU and the newer 8080 CPU, thus a microprocessor built around one of these CPU's was sought. The 8008 CPU was selected over the 8080 because of the cost differential which existed at the time of selection, plus the fact that the performance requirements of MIDAS IIA were well within the capabilities of the 8008 chip. A second factor was the ready availability of a PL/M compiler in the local computer center programs library.

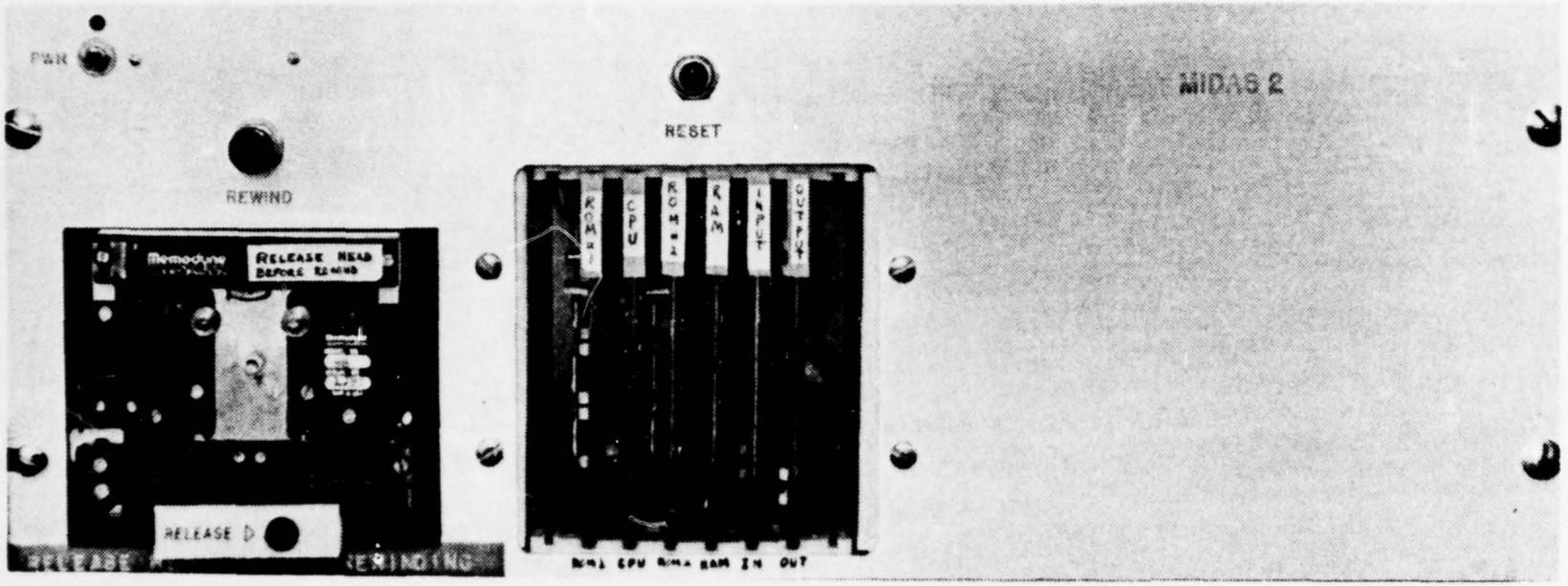


Figure 3 - MPS-805 MICROPROCESSOR

Third, an Intel Corp. microcomputer development system called the INTELLEC 8 was available locally for use in program/system development (see Fig 15). Finally, the selection of the Pro-Log 8008-based microprocessor was made because it was cheaper and more compact than Intel's equivalent, and it was available from a local manufacturer.

Operational and technical specifications of the MPS-805 Microprocessor are excerpted from Ref 2 and are shown in Tables III and IV. The MPS-805 is contained on five card modules as described below.

a. C.P.U.

The CPU card is built around the 8008 CPU chip, which also contains the Arithmetic Logic Unit (ALU). The CPU controls the internal operation of the microprocessor modules, i.e. memory, ALU, and input/output (I/O) as directed by control program instructions. The "RESET" button located immediately above the microprocessor in Fig 3 is used to cause the CPU to stop executing current program instructions and return to program address 0000, where it then proceeds to execute "restart" instructions.

Operational

CPU

Executes all of the 8008 instructions.
4 microsecond time state cycle using 8008 (MPS 1805).
2.8 microsecond time state cycle using 8008-1 (MPS 805-1).

Memory for data or program storage card expandable to any combination of ROM and RAM to 16384 words
ROM, 2048 word capacity per card.
RAM, 4096 word capacity per card.

Input and Output
Input gates implement the INP instructions.
Output latches implement the OUT instructions.

Interrupt or External Restart
Single line, synchronized interrupt on CPU card can be optionally wired for multi-level interrupt or Power-on external restart.
Multi-level Interrupt: Control lines available for external interrupt such as 8118 priority interrupt card.
Power-on and external restart option: CPU starts at instruction location 0000 by wiring restart output from CPU card to Interrupt Request input.

DMA (Direct Memory Access)
Data, address, and control lines are 3-state disconnected by the CPU following a HLT instruction allowing DMA by peripherals. The CPU must be interrupted to continue following a HALT.

Electrical Requirements
Refer to individual data sheets and schematics on the 8111, 8114, 8115, 8116, and 8117 for interface and wiring.

Power Requirements for the five card set fully loaded
+VCC = ±5% @ 3.3 Amp maximum (35mA per ROM, 50mA per RAM)
GND 0 volts
- VDD = -9 volts ±5% @ 900 mA maximum (35 mA per ROM)

Hardware
Compatible with Series 8400 interface cards.
Fits CR5, CR10 or CR19 card racks
Use M273 power supply
PROM's programmable on Series 81 programmers

Software
MPS 800 hardware is fully compatible with any 8008 software assuming I/O and interrupt can be assigned compatibly. Teletype operating system and system monitor available. Assemblers, compilers and simulators available through computer time-sharing services.

Table III. MPS-805 Operational Specifications

Table IV. MPS-805 Technical Specifications

Physical

Three 4.5" by 6.5" printed circuit cards
- One 8111 CPU card
- One 8114 Input card
- One 8115 Output card
- One 8116 ROM card
- One 8117 RAM card

Connector Requirement for each card
56 pin, 28 position dual read-out on 0.125 centers

CPU Card includes
8008 CPU
Crystal clock
Address latches, data buffers, and control decode circuits.
Power-on and external restart.
DMA buffers.

ROM Card includes
One 1702A PROM (256 bytes) and eight PROM sockets
Socket for card expansion circuit (up to 8 cards)

RAM Card includes
Eight 2102 RAM (1024 bytes) and thirty-two RAM sockets
Socket for card expansion circuit (up to 4 cards)

Input Card includes
32 TTL input selector circuits addressable in groups of 8
Socket for card expansion circuit (up to 2 cards)

Output Card includes
32 TTL output latch circuits addressable in groups of 8
Socket for card expansion circuit (up to 6 cards)

b. ROM

Read Only Memory (ROM), as the name implies, is a non-destructible type of memory device which cannot be overwritten while installed in the operating circuit card. The particular ROM devices used in MIDAS IIA are re-programmable and are thus dubbed "PROM"s. A PROM is electrically programmed using a specialized ROM programmer, and erased for reuse by exposure to short-wave ultraviolet light for 10-15 minutes. The major advantage of PROM-type memory is that it is "permanent"; i.e., it retains programmed instructions when power is removed and thus does not require re-loading every time power is applied (as RAM-type memory does).

MIDAS IIA has a fairly large control program of approximately 4,000 bytes and thus requires two Pro-Log ROM cards. Wiring of the Pro-Log card cage to include an extra ROM card is covered by Refs 3 and 4.

When this project was initiated in early 1975, PROM chips (of 256 bytes) cost $50 each, which was considered a primary disadvantage, since 16 total are required. The price has decreased steadily however, and in mid-1976 the cost was approximately $10 each. Reference 5 contains valuable information and tips on PROM use, including procedures to follow when acceptance checking new PROMs, and is a highly recommended reference for the user.

c. RAM

Random Access Memory (RAM) is a read/write type memory which can be written into and read from programmatically. It is slightly faster and cheaper than ROM-type memory but has the drawback that it is a volatile memory medium. Thus a RAM's memory contents are lost when any interruption in electric power occurs, thus it must be reloaded each time power is applied. MIDAS IIA has 2,048 bytes of RAM installed, but only about 1,000 bytes are currently used, with the remainder available for future program growth.

Table V. MIDAS IIA I/O Port Assignments

OUTPUT PORT 0	
BIT	FUNCTION
1	MUX CHANNEL SELECT 1
2	" 2
3	" 4
4	OPEN
5	CARD/FUNCTION SELECT 1
6	" 2
7	" 4
8	" 8

OUTPUT PORT 1	
BIT	FUNCTION
1	LEVEL SELECT 1
2	" 2
3	" 4
4	" 8
5	ANALOG MUX SELECT 1
6	" 2
7	" 4
8	" 8

OUTPUT PORT 2	
BIT	FUNCTION
1	TTY OUT FROM MICROPROCESSOR
2	CASSETTE TAPE START
3	OPEN
4	SAMPLE*/HOLD CMD
5	OPEN
6	"
7	"
8	"

OUTPUT PORT 3	
BIT	FUNCTION
1	CASSETTE TAPE DATA 1
2	" 2
3	" 3
4	" 4
5	" 5
6	" 6
7	" 7
8	" 8

INPUT PORT 0	
BIT	FUNCTION
1	MUX COMBINER DATA 1
2	" 2
3	" 3
4	" 4
5	" 5
6	" 6
7	" 7
8	" 8

INPUT PORT 1	
BIT	FUNCTION
1	OPEN
2	"
3	"
4	"
5	"
6	"
7	"
8	TTY IN TO MICROPR.

d. Input

The Input card, which uses positive logic levels, is configured with four "ports" of eight digital lines each, or 32 total input lines. The MIDAS IIA system is designed with a digital MUX (page 42) which presently multiplexes 16 channels (or ports) of eight lines each (expandable to 32) all into one of the Pro-Log input ports. Also the Analog-to-Digital Converter (ADC) module multiplexes 16 analog signals into two of the MUX channels, thus the actual input capacity of MIDAS IIA is quite large. Table V contains details of individual I/O port assignments for MIDAS IIA.

e. Output

The Output card is configured with four ports of eight digital lines each or 32 total output lines. Program output values are "latched" into the designated output ports and remain valid (set) until the system is reset or the program changes the output data to the port. Outputs are referenced to positive logic levels, as are inputs. Table V contains details of individual I/O port assignments for MIDAS IIA.

2. Level Select Unit

a. General

The Level Select Unit, shown in Fig 4, multiplexes five quartz thermistor sensor probes to the Quartz Thermometer instrument and four relative humidity/temperature sensor probes to the Digital II instrument. This feature of MIDAS IIA allows a single precision instrument to be used in the collection of high quality data from several different locations at a minimum of cost.

Sensor probes are connected to "levels" which are programmatically selected by the microprocessor. Manual selection of a level can be made by using the appropriate momentary-hold toggle switch located on the level select face panel. Level zero represents sea water temperature and has only a quartz temperature sensor attached, but levels one through four have several sensor types attached. The MIDAS IIA system has been designed throughout for eventual expansion to six levels, thus the Level Select Unit also has connectors, switches, and indicators for levels five and six.

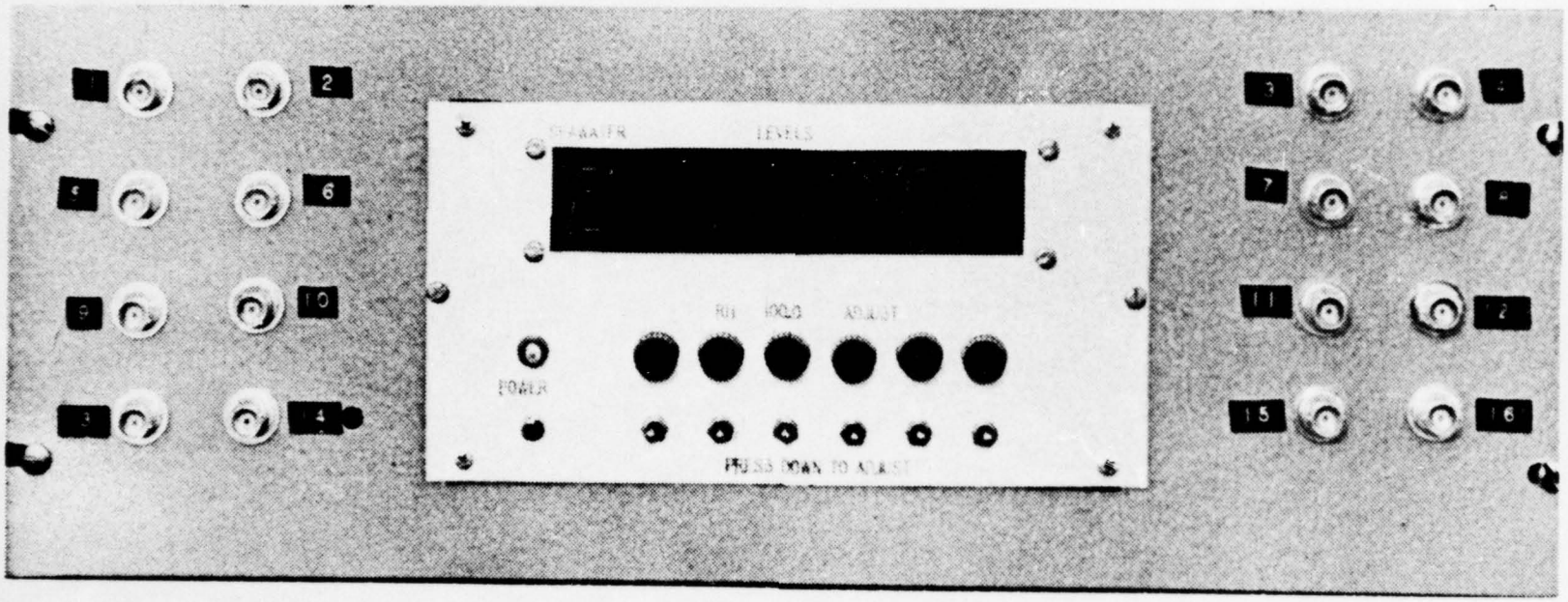


Figure 4 - LEVEL SELECT UNIT

b. Calibration of relative humidity

The front panel of the Digital II instrument (Fig 8) contains digital displays for relative humidity and temperature plus several lever-type switches to control operating modes. One of the lever switches is marked ADJ. and is used when adjusting the full scale value of relative humidity to read 100%. To properly adjust all relative humidity sensors connected to the Digital II through the Level Select Unit, the following procedures are required.

First the ADJ lever on the Digital II face panel is selected. Then on the Level Select Unit panel (Fig 4) the left-most momentary-hold toggle switch located beneath the digit for level #1 is pressed down. This action causes the level indicator number "1" to illuminate and the corresponding level's humidity/temperature sensor to be interconnected to the Digital II unit. While holding the switch down, the humidity reading on the Digital II panel is observed and adjusted for 100% by twisting the black knob located immediately above the toggle switch being held. Each level's humidity sensor is set by pressing the appropriate toggle switch and adjusting in the same manner as indicated above for level one.

3. Digital Clock

The digital clock (Fig 5) used in MIDAS IIA is an "off the shelf" standard design circuit, giving an hour-minute-second Light Emitting Diode (LED) display and corresponding BCD outputs. Referring to Fig 5, note that toggle switches are provided for setting the clock by advancing the time either fast or slow or holding the current value. The clock circuit, contained on the card visible at the left of Fig 5, is easily accessible for maintenance purposes. Notably, the clock has proven to be highly reliable.

The BCD output data for time is connected to MIDAS IIA digital MUX channels 00 and 01. The clock circuit sends only a single display digit in BCD format to MUX channel 01 and uses channel 00 to identify which digit is being sent. It cycles through the six digits starting from the least significant digit (seconds) to the most significant digit (10's of hours) continuously. MIDAS IIA programmatically reconstructs the time value within the microprocessor (see page 60 for details) and uses this time for the program reference clock. Details of clock data input format to the MUX are contained in Fig 11.

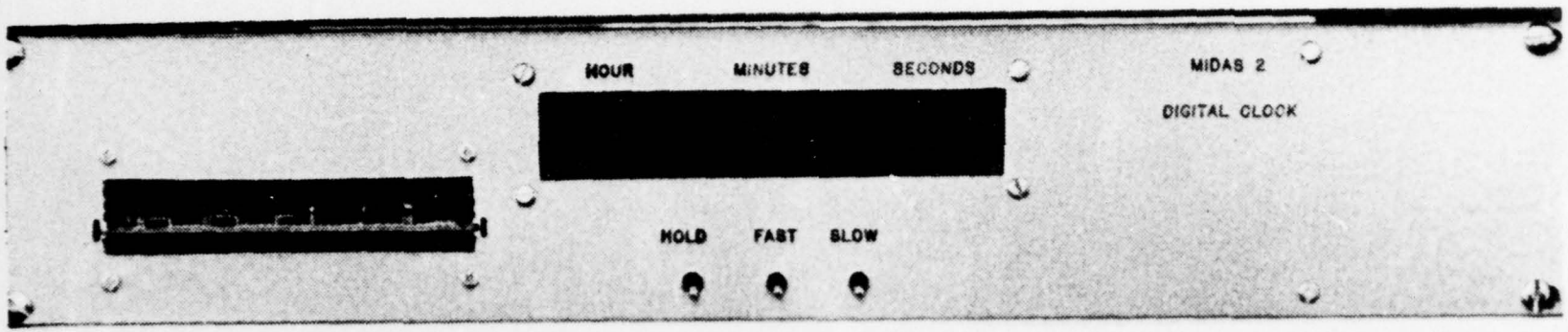


Figure 5 - DIGITAL CLOCK UNIT

4. Barometer

The Validyne Model DB-99 Digital Barometer, as shown in Fig 6, is not housed in the MIDAS IIA equipment cabinet but is positioned conveniently close and connected via ribbon cable to MIDAS IIA digital MUX channels 02 and 03. It produces standard binary coded decimal (BCD) digital output data which spans the range of 25.00 thru 31.00 inches of mercury. The data lines consist of fourteen data bits and one status bit. Details of barometric data input formatting appear in Fig 11. Reference 6 contains detailed information concerning instrument operation and wiring pin connections.

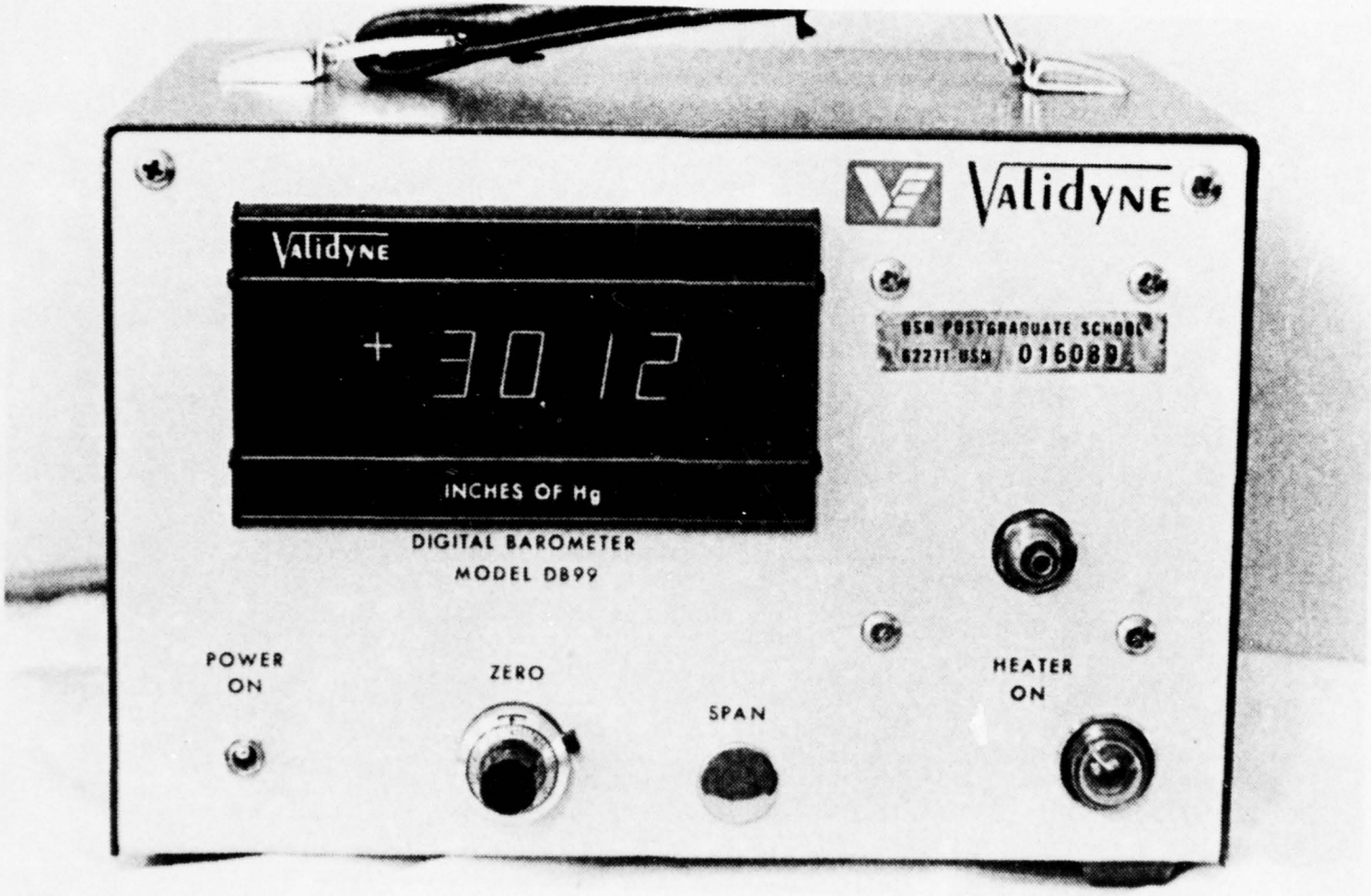


Figure 6 - VALIDYNE DIGITAL BAROMETER

5. Quartz Thermometer

The Hewlett-Packard Model 2801A quartz thermometer, shown in Fig 7, produces a digital output which spans the temperature range of 00.00° thru 49.99° Celsius. Fifteen data bits and one status bit are outputted to MIDAS IIA in an inverted logic form and are connected to digital MUX channels 12 and 13 via the Quartz Buffer card. The outputted digital data are encoded in a 4-2-2-1 format as shown in Table VII. MIDAS IIA programmatically inverts the data and converts then to hexidecimal form for storage in microcomputer memory.

The Quartz Thermometer face panel (Fig 7) has a display readout and several control selectors. For proper operation, the resolution scale should be set to ".01", the input mode to "T1", and the sensitivity adjustment rotated counter-clockwise.

Note: For proper MIDAS IIA timing, the display interval must be set for a sample rate of at least once every second as indicated by the flashing of the light next to the control knob.

The input data format fed to MUX channels 12 and 13 is described in Fig 11. Reference 7 contains details concerning physical and technical specifications of the unit.

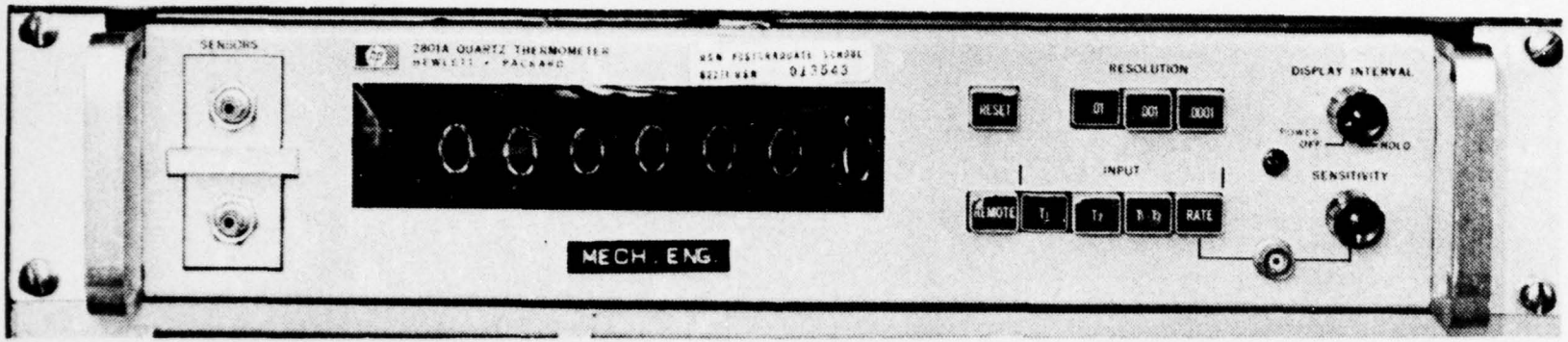


Figure 7 - HEWLETT-PACKARD QUARTZ THERMOMETER

6. Relative Humidity/Temperature Indicator

The Hygrodynamics Model Digital II relative humidity/temperature instrument, Fig 8, produces two independent sets of BCD coded digital outputs representing relative humidity and temperature respectively. Its operating ranges are 00.0% to 99.9% relative humidity and 00.0° to 99.9° Celsius or Fahrenheit as selected. The twelve data bits and one status bit for relative humidity are connected to MIDAS IIA digital MUX channels 15 and 16, while the temperature data and status bits are connected to MUX channels 13 and 14. Details of relative humidity and temperature data input formatting appear in Fig 11. Procedures for calibrating relative humidity are located on page 29. Complete information on the physical and operational specifications of the Digital II unit may be found in Ref. 8.

7. Digital Cassete Tape Recorder

The Memodyne Corporation Model 171 Magnetic Tape Recorder records digital data incrementally on industry standard "PHILLIPS" data cassettes. The write-only recorder, pictured on the left of Fig 3, features eight-bit parallel input from the microprocessor but writes data on tape serially. Data are written at the rate of 100 bytes per second with a density of 320 bits per inch. The advertised capacity of a 300-foot data cassette is 144,000 bytes of information, which equates to 65 plus hours of MIDAS IIA continuous operation at the max output rate of one minute averaging intervals.

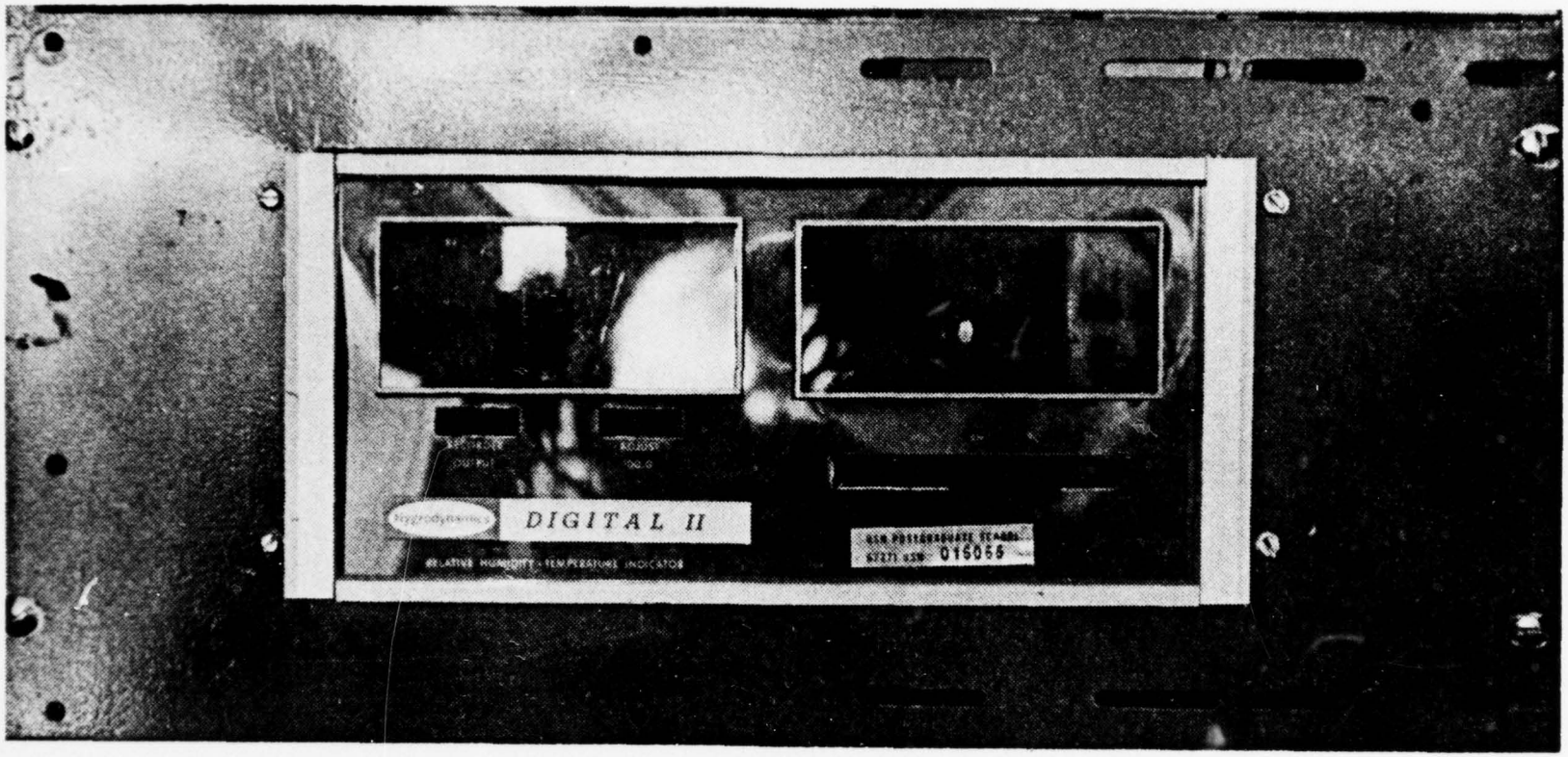


Figure 8 - HYGRODYNAMICS RELATIVE HUMIDITY/TEMPERATURE UNIT

Once operation is initiated, cassette recording of MIDAS IIA output data is fully automatic. Procedures for proper cassette initiation are given in Table IX. During operation, the data stored on cassette tape by MIDAS IIA are virtually the same as that outputted on the Teletype and thus provide a redundancy factor to insure that a permanent output record is made. Additionally, the cassette may be used for inputting MIDAS IIA output data to the HP-9830A for data analysis, as indicated in Section IV and Appendix A.

Details of interface wiring appear in Appendices B, C, and D. Complete recorder specifications and schematics are contained in Ref. 9.

8. ASR-33 Teletype

The Teletype Corporation Model ASR-33 Teletype, Fig 9, is presently the primary input/output interface device for MIDAS IIA. It is connected, for a full duplex mode of operation, to the microprocessor via the TTY Buffer card as explained on page 46. Technical details concerning the ASR-33 are found in Ref. 10. The microprocessor converts to and from ASCII for output/input. The ASR-33 has an associated paper tape punch which is normally selected during system operation to produce a permanent output record of the "hard copy" printed page. This paper tape may then be used to obtain a data deck of Hollerith-punched cards as explained in Section IV A, or inputted to the HP-9830A for data analysis as indicated in Section IV C and in Appendices E and F.

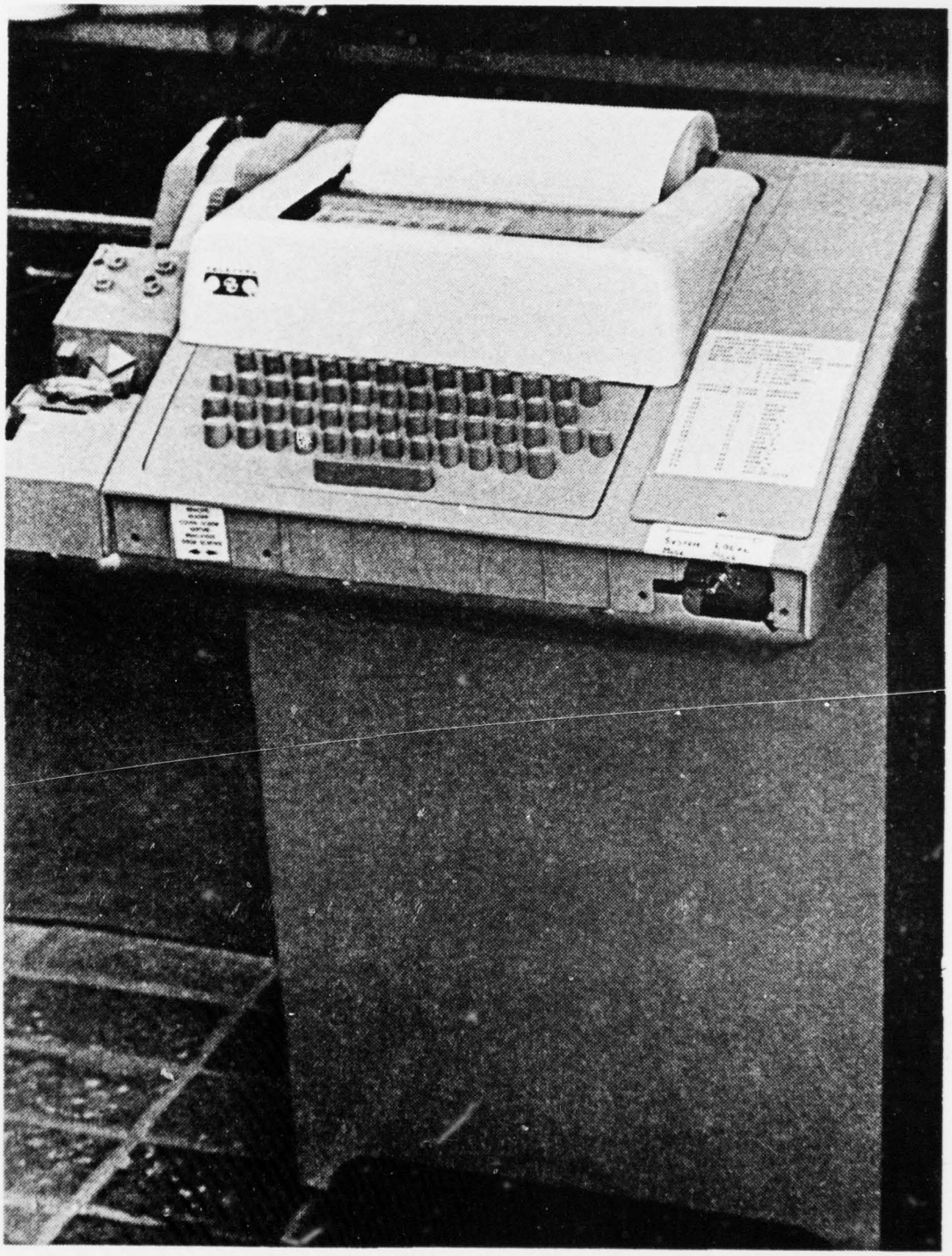


Figure 9 - ASR-33 TELETYPE

Selection of the Teletype for a data output device was originally based on its low cost and local availability. It has a serious drawback, however, due to its slow data output rate. For example, the present four-level data output cycle, as shown in Table XII, takes approximately 22 seconds. This lengthy time requires that a minute's delay between data-gathering intervals be inserted to allow for output. The replacement of the TTY is discussed in this report under the Conclusions and Recommendations section.

9. Power Supply Requirements

The power supply for MIDAS IIA is located at the bottom of the equipment cabinet shown in Fig 2. The supply produces various D.C. voltages for use by numerous components of the system as outlined in Table VI.

Table VI. MIDAS IIA Power Supply Requirements

VALUE	UNIT(S) SUPPLIED
+5 V.D.C.	MPS-805 MICROPROCESSOR
	ALL UTILITY CIRCUIT CARDS
	MEMODYNE CASSETTE RECORDER
+12 V.D.C.	MEMODYNE CASSETTE RECORDER
+15 V.D.C.	ANALOG MULTIPLEXER
	SAMPLE AND HOLD MODULE
	ANALOG-TO-DIGITAL CONVERTER
-10 V.D.C.	MPS-805 MICROPROCESSOR
	TTY/BUFFER CARD
	DIGITAL CLOCK
-15 V.D.C.	ANALOG MULTIPLEXER
	SAMPLE AND HOLD MODULE
	ANALOG-TO-DIGITAL CONVERTER

All remaining equipments (mostly instruments) are powered by standard +110 V.A.C. 60Hz.

10. Utility Circuit Cards

a. Digital Multiplexer (MUX)

MIDAS IIA uses a discrete digital multiplexing input scheme to permit a large number of eight-bit channels (or ports) to feed a single eight-bit input port of the MPS-805 microprocessor. This is an acceptable procedure for this application since MIDAS IIA is a "mean averaging" system which takes data samples infrequently (once every 30 seconds) and at a moderately slow scanning rate. The utility card cage, Fig 10, shows MUX cards #0 and #1 installed at the extreme left with slots left open for cards #2 and #3.

All MUX cards are identically wired as per Appendix G. Basically, each card has eight input channels of eight bits each which are multiplexed to a single card output channel. Three channel address lines come from the microprocessor's output port #0 (lower) thru inverters (on the TTY/Buffer card) and connect to all cards (presently cards #0 and #1). These channel address lines allow the microprocessor to select a single input channel on each card. In order to enable only one card's input channel and inhibit the other cards, a card select line is provided on each MUX card. The microprocessor outputs a card select address on output port #0 (upper) to a decoder chip on the TTY/Buffer card, which in turn selectively enables only the MUX card desired.

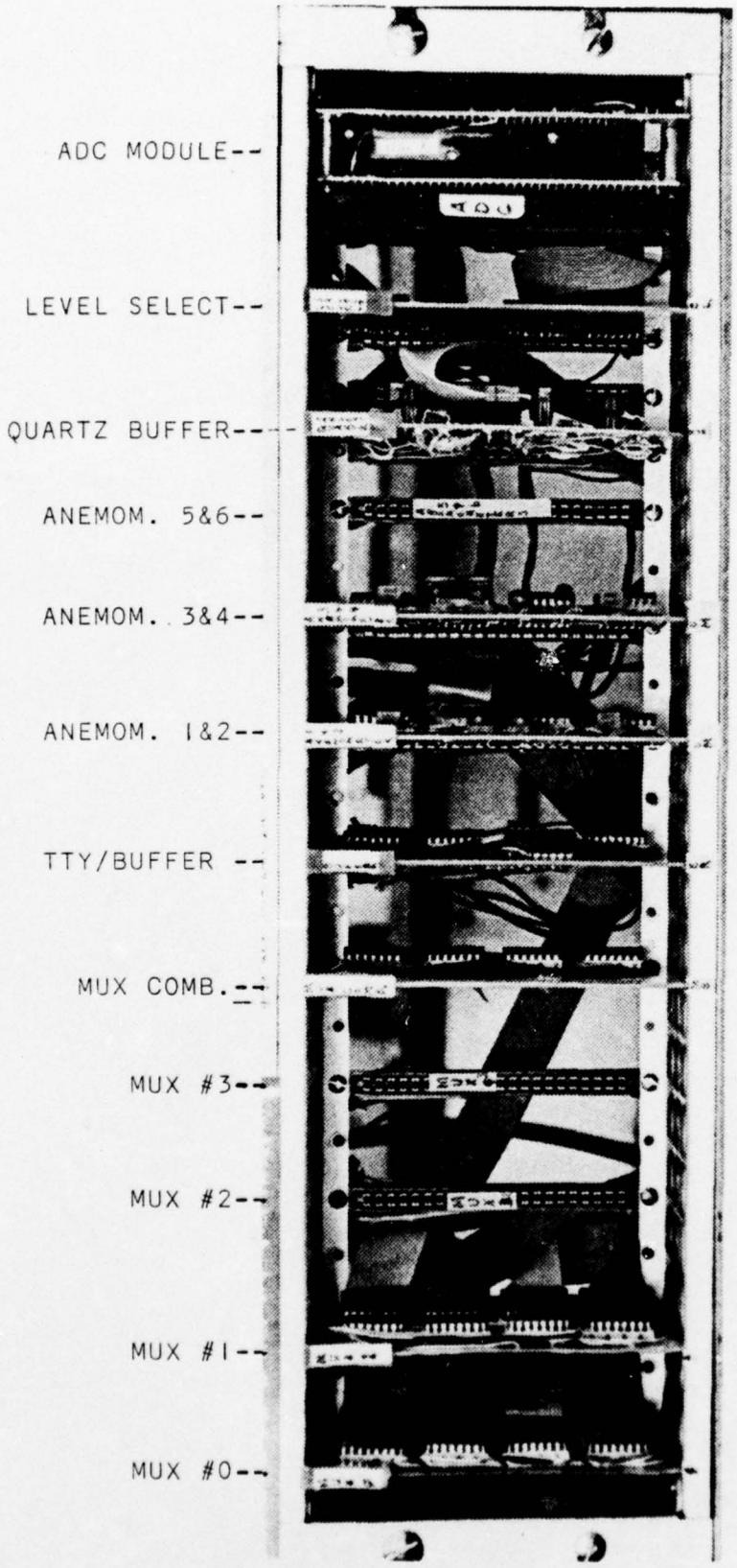


Figure 10 - MIDAS IIA UTILITY CARDS

The above addressing scheme permits the microprocessor to programmatically select any available input channel by setting a two-digit value on output port #0, where the upper digit represents the MUX card number (0-3) and the lower digit indicates the channel number (0-7).

At present, MIDAS IIA has channels 00-07 and 10-17 installed, with wiring for 20-27 installed and provisions for adding 30-37 for a total of 32 input channels possible. Details of MUX card wiring are contained in Appendix G. Figure 11 contains information concerning the various MUX input data formats presently installed.

MUX CHAN	FUNCT.	FORMAT UPPER				FORMAT LOWER				PURPOSE UPPER	PURPOSE LOWER	NOTE
00	CLOCK	X	X	HIO	HI	MIO	MI	SIO	SI	HOUR INDICATORS	MIN/SEC INDICATORS	1
01	CLOCK	X	X	X	X	D	D	D	D	BLANK	ONE BCD DIGIT	1
02	BAROM.	VI	X	D	D	D	D	D	D	10's inches Hg.	Inches Hg.	
03	BAROM.	D	D	D	D	D	D	D	D	Inches Hg./10	Inches Hg./100	
04	ANEMOM 1	B	B	B	B	B	B	B	B	HEX DIGIT	HEX DIGIT	
05	ANEMOM 2	B	B	B	B	B	B	B	B	HEX DIGIT	HEX DIGIT	
06	ANEMOM 3	B	B	B	B	B	B	B	B	HEX DIGIT	HEX DIGIT	
07	ANEMOM 4	B	B	B	B	B	B	B	B	HEX DIGIT	HEX DIGIT	
10	ADC	VO	X	X	X	X	B	B	B	VALID DATA INDIC.	BINARY FRACTION #	
11	ADC	B	B	B	B	B	B	B	B	BINARY FRACTION #	BINARY FRACTION #	
12	QUARTZ	VI	D	D	D	D	D	D	D	10's of DEGREES	UNITS of DEGREES	2
13	QUARTZ	D	D	D	D	D	D	D	D	DEGREES/10	DEGREES/100	2
14	H.TEMP	VO	X	X	X	D	D	D	D	VALID DATA INDIC.	10's of DEGREES	
15	H.TEMP	D	D	D	D	D	D	D	D	UNITS of DEGREES	DEGREES/10	
16	H.HUM.	VO	X	X	X	D	D	D	D	VALID DATA INDIC.	10's of % R.HUM.	
17	H.HUM.	D	D	D	D	D	D	D	D	UNITS % of R.HUM.	% R.HUM./10	

KEY: B = Binary; D = BCD Digit; H = Hours; M = Min.; S = Sec.; V = Validity; X = Ignore

NOTES:

1.Only one of the hour/min/sec indicator bits will be set at any time. A set bit in a position indicates the value of the BCD digit simultaneously inputted on Chan. 01.
2.Data is inverted and encoded in 4-2-2-1 code.

Figure 11 - DIGITAL MUX INPUT DATA FORMATS

b. Mux Combiner

The MUX Combiner card does just what its name implies, namely it takes the eight-bit input channel data coming from any of the MUX cards (a specific card/channel is selected by the microprocessor) and passes them on to input port #0 of the microprocessor. Details of MUX combiner card wiring are contained in Appendix H.

c. TTY/Buffer

The TTY/Buffer card provides a full duplex current-loop interface with the ASR-33 Teletype, plus it inverts and decodes various microprocessor control/address outputs. Details of TTY/Buffer card wiring are contained in Appendix I.

d. Anemometer Counters

The Anemometer Counter card contains binary counter circuits which record the number of revolutions made by a cup-type anemometer wind velocity sensor. The system presently has two counter cards installed with wiring for a third card completed (refer to Fig 10). The cards are identical, with each card containing two sets of eight-bit binary counters for handling two anemometer sensor inputs. The output of each counter is connected via ribbon cable to a MUX input channel (currently channels # 04, 05, 06, and 07 are used). Details of MUX data input format appear in Fig 11. All of the counters on all cards have their reset/count control lines tied together and connected to an output line of the decoder chip on the TTY/Buffer card.

That decoder is in turn addressed by output port #0 (upper) of the microprocessor, to cause a simultaneous reset of all counters by program control. Use of eight-bit binary counters restricts the maximum number of recorded counts (cup revolutions) possible to 255 decimal, but since the anemometers are scanned and reset every 30 seconds, this is not a practical limiting factor. Appendix J contains anemometer counter card wiring information.

e. Quartz Buffer

The Quartz buffer card takes digital data pulses coming from the quartz thermometer (discussed on page 34) and performs level shifting to convert the pulses to TTL voltage standards. The TTL values are then connected via ribbon cable to MUX channels 12 and 13. Details of MUX input data format appear in Fig 11.

f. Level Select

The Level Select card takes microprocessor output addressing from output port #1 (lower) and feeds it to a decoder chip which pulses the appropriate output line as indicated by the address selected. The output pulses are inverted and then connected to the Level Select Unit (described on page 27), thus enabling the microprocessor to programmatically control level selection. Details of wiring are found in Appendix K.

g. Multiplexed Analog to Digital Converter (ADC)

MIDAS IIA is designed to accept up to 16 analog voltage input signals in the range of 0 to 5 volts D.C.

This is accomplished by multiplexing analog signals into a sample and hold module which is in turn connected to an Analog-to-Digital Converter (ADC) module as shown in the functional schematic, Fig 12. The assembled MIDAS IIA ADC unit is pictured in Fig 13.

Analog inputs to MIDAS IIA are made by connecting a source signal to one of the BNC connectors located on the Level Select Unit panel (Fig 4). The 16 BNC connectors are arranged on the panel such that each vertical column of four connectors corresponds with a column of data output on the TTY printout (see Table XII for a sample printout). Signals are fed to a Datel Corporation Model MM-16 Analog Multiplexer, Ref 11, which gates one of the signal channels, selected by microprocessor output port #1 (upper), through a buffer amplifier and to the sample and hold module.

The Datel Model SHM-4 Sample and Hold module, Ref 12, is designed to sample and track an analog signal coming from the multiplexer and either output it as received, or "freeze" the output value at some instant and hold that value at it's output. It is controlled by a single command line (sample/hold) connected to microprocessor output port #2. The sample and hold module is required to permit a rapidly changing analog input signal to be properly converted by the ADC module. The ADC module uses a dual-slope integration technique for analog-to-digital conversion which takes approximately five milli-seconds to perform a convert cycle. During this time interval, the input to the ADC must be held steady to insure an accurate conversion. This is accomplished by programmatically "freezing" the signal with the sample and hold module just prior to ADC conversion.

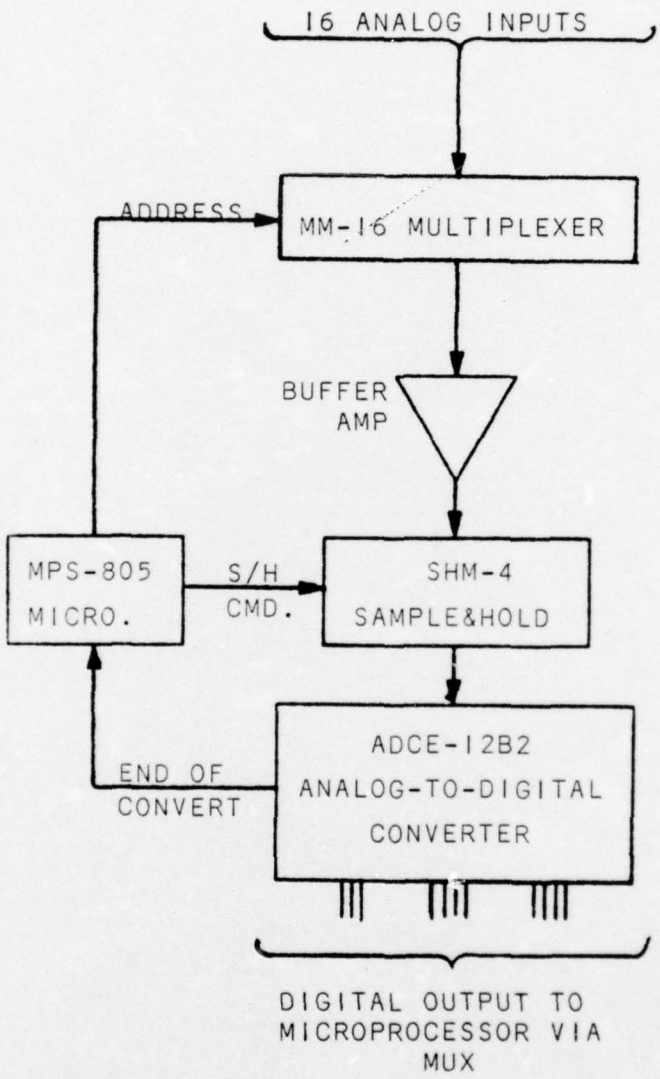


Figure 12 - ANALOG TO DIGITAL CONVERTER (ADC) SCHEMATIC

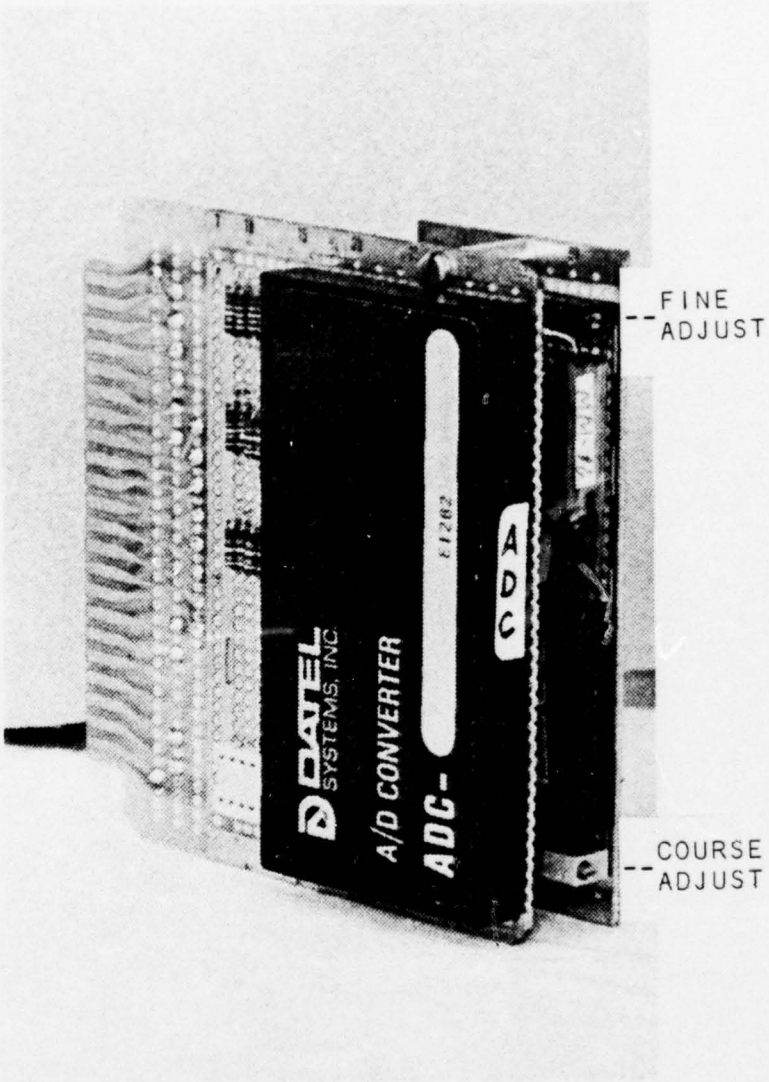


Figure 13 - ANALOG TO DIGITAL CONVERTER (ADC) CARD

The output of the sample and hold module is fed to the ADC module via a voltage divider. The voltage divider is required because the ADC module has an input voltage range of 0 to 1 volt D.C. and the desired analog signal input range is 0 to 5 volts D.C. Thus a 5 to 1 voltage divider was constructed using a 10K Ohm potentiometer.

Scaling down the voltage to the ADC module results in little error within the input voltage range of approximately 0.2 to 4.8 volts, however operation near the extreme limits of 0 and 5 volts is unreliable. This error is due in part to the fact that the ADC output digital value is multiplied by a factor of five for use in the computer. This scaling down and back up again thus limits the accuracy of ADC signals to approximately plus or minus .02 volts. Therefore, it is recommended that a properly scaled ADC module be incorporated during some future system update.

The output of the voltage divider is connected to the analog input of the Datel Corp. Model ADCE-12B2 Analog-to-Digital Converter (Ref 13). The ADC module has a built-in oscillator which is used to trigger the convert cycle every 8 milli-seconds continuously. The output data are in 11-bit binary form, representing the value of 0 to 1 volt. Validity of output data is indicated by an "end of convert" status line from the ADC module. The end of convert line and the 11 data lines are connected via ribbon cable to MUX input channels 10 and 11. The format of input data to the MUX is covered by Fig 11. Details of ADC card wiring appear in Appendix L.

C. MIDAS IIA CARD INTERCONNECTIONS

The utility circuit cards are mounted in a card cage as shown in Fig 10. The operation and wiring of the individual cards was covered in the previous section. Wiring interconnections between the utility cards and other units are detailed in Appendix M. The MPS-805 microprocessor's Input/Output ports are functionally assigned as shown in Table V and are wired into the system as per Appendix N.

D. MIDAS IIA SOFTWARE PROGRAM ORGANIZATION

1. General

The software program for MIDAS IIA was written in PL/M, a "high level" language developed by Intel Corporation to support its microcomputer product line. One of the major advantages of using PL/M is that it facilitates the writing of a self-documenting program which makes future changes or additions a simple task. The MIDAS IIA PL/M program, listed in Appendix O, was written in a modular form known as "top down structured programming" wherein many subroutines (called procedures in PL/M) are used. This technique allowed the executive (or main) program to be developed and debugged more rapidly because many of the detailed tasks were placed in procedures. Initially, most of the procedures were mere "dummies" that were declared but did nothing when called but return control to the executive. Once the executive was developed, then the

procedures were added as they were developed and debugged. This allowed a step-by-step controlled growth of the program until all the dummy procedures were finally replaced with functioning ones.

2. Memory Organization

MIDAS IIA has two types of memory as explained previously. The program instructions and TTY messages are stored in the permanent area called ROM while the sampled data values and computed averages are stored in RAM. Memory address numbering is organized such that the low numbers (0000H to 0FFFH where H indicates a Hexidecimal number) are assigned to ROM while those from 1000H upward designate RAM. A diagram of the MIDAS IIA memory organization is shown in Fig 14.

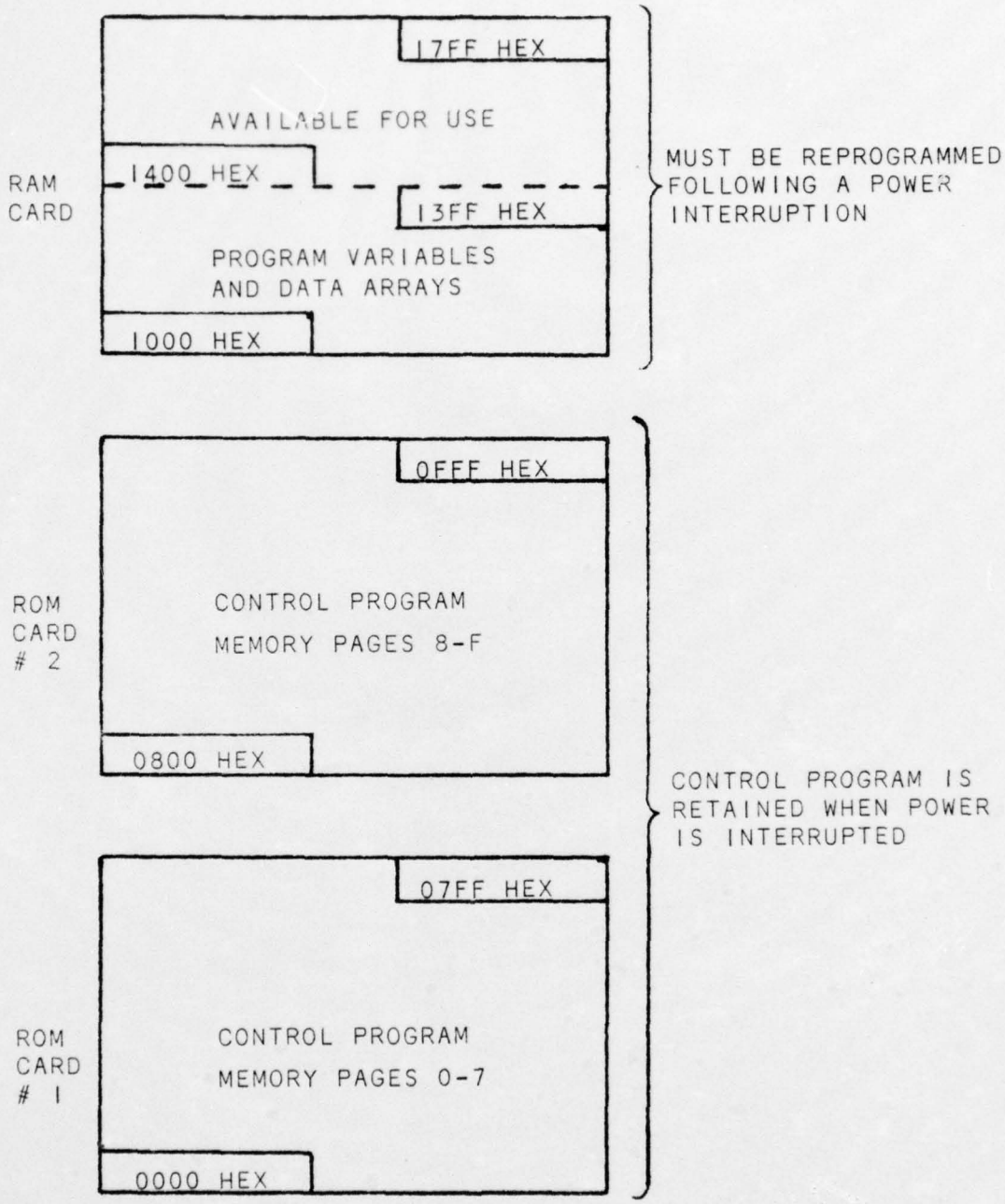


Figure 14 - MIDAS IIA MEMORY MAP

3. PL/M Program Procedures

The PL/M procedure descriptions that follow are arranged in the order in which they appear in Appendix O. These descriptions are designed to provide insight into the basic functions performed by the procedures without getting into the details of program coding. Users interested in program coding are referred to Appendix O along with Refs 14 and 15.

a. BCDTOHEX

This procedure accepts a 2-digit binary coded decimal (BCD) number (a byte value variable called BCD) and converts it to a hexidecimal byte value, which is then returned to the calling program. The maximum allowed input BCD value is 99 decimal. This procedure is required because several of the sensors produce BCD outputs but the microprocessor works in hexidecimal (an extension of binary numbering).

b. PULSE

This procedure repeatedly samples microprocessor input port #0 (the data input port multiplexed from the digital MUX) and checks for the occurrence of a pulse on a selected data line. The line is selected by using the byte parameter PULS$MASK to logically mask off the desired line. The pulse is then checked by comparing the masked line with the byte parameter MATCH$VAL for a match, then for a mismatch. This procedure is useful for checking the "data valid" status line from various sensors to insure that the

sensors' data lines are valid before sampling them. Proper use of this procedure is to provide a pulse mask byte parameter called PULS$MASK which isolates the sensor's status line, and a byte parameter called MATCH$VAL which represents the valid data status condition. This procedure is required because the possibility exists that a sensor could be checked and found valid but it is just about to go invalid for another convert cycle. Thus, by the time the data lines are checked they might be invalid. By waiting for the current valid cycle (if present) to complete then waiting for the invalid cycle to complete, there will be plenty of time to sample the data and be assured of its validity.

c. ANEMOMETER

This procedure scans the four cup anemometer counters and adds their numerical values into the appropriate VALUE memory array element values. Upon completing the scan, a command is issued to reset all the counters to zero. Eight-bit binary counters are used in MIDAS IIA thus no number system conversion is required.

d. DELAY

When called, the DELAY procedure implements a variable program delay of up to 20 milliseconds by accepting a byte parameter called AMOUNT whose value is between zero and 255 decimal. DELAY in turn calls a PL/M function called TIME which does the actual delaying as explained in Ref 14.

e. LEVL$DELAY

This procedure produces a variable time delay of up to 5 seconds by accepting a btye parameter called LNGTH whose value is between zero and 255 decimal. The DELAY procedure is then called in a loop which is controlled by the number in LNGTH. This procedure is required to produce a three second program delay when changing levels on the Level Select Unit, to allow the Quartz Thermometer sensor to settle before continuing sampling.

f. QUARTZ

This procedure samples a quartz sensor input, inverts it, converts the 4-2-2-1 data format code to hexidecimal, and adds the numerical value into the appropriate VALUE memory array element value. Two nested procedures called SHIFTER and CONVERT are used by QUARTZ to make the conversion from 4-2-2-1 to Hex (or binary) as per Table VII.

Table VII. 4-2-2-1 Code Conversion To Binary

1. The 4-2-2-1 code is an obsolete bit coding format which was used by some manufacturers prior to the general industry-wide acceptance of BCD coding. The Hewlett-Packard Quartz Thermometer used in MIDAS IIA produces this digital output format thus necessitating a special conversion to binary as shown below. Conversion is performed through the use of software programming in the "QUARTZ" procedure.

NUMBER	4-2-2-1	BINARY
1	0001	0001
2	0010	0010
3	0011	0011
4	0110	0100
5	0111	0101
6	1100	0110
7	1101	0111
8	1110	1000
9	1111	1001
0	0000	0000

g. MULTI$SENSOR

Because of the four parameters passed to this procedure, it is flexible enough to handle four different sensor types (barometer, humidity, H Temp., and ADC) with the same code. Functionally, it samples the sensor type identified by the byte parameter NAME, converts from BCD to Hex if required (all except ADC), and adds the numerical value into the appropriate "VALUE" memory array element value. The parameters PULSS$MASK and MACH$VAL are used to call the PULSE procedure as discussed previously. The UPPERWORD$MASK parameter is used to mask off the valid data bits in the upper word of data input.

h. TYCHAR$IN

When called, this procedure looks at the data line from the TTY to microprocessor input port #1, and waits until a TTY serial ASCII character (American Standard Code for Information Interchange) input is sensed. It then converts the character from serial to parallel, inverts it, strips off the ASCII parity bit, and returns the ASCII character value to the calling program.

i. TTYCHAR$OUT

This procedure accepts a byte parameter value called CHAR (assumed to be an ASCII coded character), inverts it, converts from parallel to serial, and outputs it serially to the TTY on microprocessor output port #2.

j. TTY$CRLF

As its name implies, this procedure is designed to output carriage return and line feed commands to the TTY. This is accomplished by successively calling the TTYCHAR$OUT procedure with the appropriate ASCII values as contained in the PL/M program "literals".

k. CLOCK

The MIDAS IIA digital clock is sampled by this procedure and a logical variable value of FF HEX is returned if the seconds value is equal to 00 or 0F HEX if it equals 30 seconds (otherwise a "False" value of 00 is returned). Also the global address variable HM is updated with the current BCD time in hours and minutes each time CLOCK is called. The digital clock is connected to MUX channels 00 and 01. The clock circuit sends only a single BCD digit at the time to MUX channel 01 and uses channel 00 to identify which digit is being sent. The CLOCK procedure cycles through and samples the six digits starting from the least significant digit (seconds) to the most significant digit (10's of hours) and then reconstructs the time in hours-minutes in HM and checks the seconds for a value of 00 or 30. This procedure is used to obtain the value of the digital clock for system timing purposes. For example, the 30-second sample interval is based on the value of the logical variable returned, and the HM variable is outputted to the TTY to mark the time of an output cycle.

l. MESSAGE

This procedure is used to output initialization messages on the TTY to the operator during the initialization procedure as discussed in Table X. Eight different messages are stored in ASCII form in permanent ROM memory and are identified to this procedure by passing the first address of one of these messages. The first byte of each of these permanent messages gives the number of characters in the following message. This value is used to tell this procedure how many characters to output to the TTY via the TTYCHAR$OUT procedure.

m. MAG$TAPE

This procedure accepts a hexidecimal byte value parameter called WORDVAL, inverts it, and outputs it to the Memodyne cassette recorder. Since the recorder requires at least 12 milliseconds between writing data bytes, there is a delay incorporated in the procedure to protect against the possibility of this procedure being recalled immediately.

n. TTY$BYTEOUT

The byte value parameter OUTVAL is passed to this procedure as a hexidecimal byte where it is split into half, then each half is encoded as an ASCII character and output to the TTY via the TTYCHAR$OUT procedure.

o. PRINTOUT

This procedure accepts a byte value parameter called INDEX which is the index number for an element of the program ANSWER address array (Table XI) to be printed out. The procedure splits the address-sized value of ANSWER(INDEX) into two bytes called TOP and BOT after first converting the value into BCD. The BCD value in TOP is then printed on the TTY as the top two digits of an answer via the TTY$BYTEOUT procedure followed by the lower two digits from BOT. Additionally, the BCD values in TOP and BOT are recorded on cassette tape using the MAG$TAPE procedure.

p. SPACES

As is implied by it's name, procedure SPACES outputs to the TTY (via the TTYCHAR$OUT procedure) the ASCII value to cause a blank space on the printout. The byte value parameter called NUMBER which is passed to this procedure determines how many spaces are outputted.

q. TTY$BITEIN

When called, this procedure waits for numerical characters to be typed in at the TTY (via the TYCHAR$IN procedure). When a character is received, it echoes it back out to the TTY via the TTYCHAR$OUT procedure, strips the ASCII coding off, and stores the digit in the lower half of the variable WORD. When another character is received, it again echoes and decodes it and stores it in the lower half of WORD after first shifting the number already there

to the upper half of the byte. If a third character is then received, the variable WORD will throw away the first digit received and keep the last two numbers. This process continues as long as valid numerical characters are typed into the TTY. If at any time during execution of this procedure, a non-hexidecimal character is typed, the procedure will echo a question mark (?), zeroize the variable WORD, and start from the beginning of the procedure. Once the two desired digits are typed in, the procedure is exited by pressing the TTY space bar. This causes the byte variable value in WORD to be returned to the calling program.

r. RESPONSE

Calling this procedure causes the TTY to print out the message (via the MESSAGE procedure) 'Y' or 'N' :. This is a query to the operator to type in either a "Y" or an "N" representing yes or no. This procedure checks the value typed in (via the TYCHAR$IN procedure), echoes it to the TTY, and returns to the calling program a logical value of "true if Y is input or a "false" if N is typed. If the operator inputs anything other than Y or N, the procedure prints out a question mark (?) and returns to the start of the procedure where it starts over with the Y/N query.

s. AVERAGE

Functionally this procedure takes the value of a sensor's sampled data memory location, which contains the sum of several data samples, and computes the mean data average by dividing thru by the number of samples taken. It converts this average to a BCD value and returns it to the calling program. Specifically, memory storage for sensor

data is contained in the address variable array called VALUE. Sensor input values are split into upper and lower halves and stored in adjoining elements of the VALUE array. These two address elements are then kept separate during the data-gathering cycle. At the conclusion of a data-gathering cycle, the adjoining VALUE elements contain summations of upper-half data samples and lower-half data samples respectively. An advantage of this method is that the address-sized elements of VALUE can store the summation of several hundred samples before exceeding their storage capacity. When this procedure is called, a byte parameter called INDEX is supplied to identify which elements of the VALUE array are to be averaged and returned.

t. ADC$PRINT

This procedure takes the sampled summations for an ADC input as stored in the VALUE address memory array, computes the mean data average, and converts the value to a BCD whole number which is stored in the appropriate element of the ANSWER array (Table XI). The procedure then calls the PRINTOUT procedure with the index of the ANSWER array element just filled. This procedure appears somewhat similiar to the AVERAGE procedure but is actually very different. It is required because the ADC converter (explained on page 47) produces an 11-bit binary fractional output value for the voltage scale of 0 to 1 volt. The fractional binary value is converted to a whole BCD number in this procedure by adding the weight of each bit into the address variable NUMB. Finally the total in NUMB is scaled up by the same amount that the real input was scaled down (by the voltage divider) by multiplying NUMB by the literal MAXADCVOLTS which is currently defined as 5.

u. DUMPLINE

This procedure is called at the end of each line of TTY output. It's purpose is to output to the TTY (via the TTYCHAR$OUT procedure) the control character "XOFF" (whose ASCII number is 13 HEX) and a blank space (ASCII number 20 HEX). These characters do not appear on the printed TTY copy, but do get punched into the paper tape copy. These two extra characters are control characters used by the timesharing computer system CP/CMS (Ref 17) to denote the end of a punch-card record. By inserting them in the MIDAS IIA output paper tape, this tape can be read into the CP/CMS system and a deck of Hollerith punched cards made, as explained in Section IV A.

4. PL/M Executive (Main) Program

The following functional description of the MIDAS IIA PL/M executive program will step through the program as listed in Appendix O. When the "Reset" button above the microprocessor is pressed, the program goes to the label "RESTART" and commences the initialization sequence as detailed in Table X. The label "CHG$LOOP" marks the block of code that implements the scanlist modification steps given in Table X. After changes are made, the program start time is entered by the operator and stored in the variable START$TIME followed by the desired averaging interval (in minutes) which is stored in the variable INTERVAL.

Once the initialization as per Table X is completed the program goes to the label "STRT$SAMPLE" and starts repeatedly checking the digital clock value for a match with

the start time entered by the operator. When a time match (including 00 seconds) occurs, the program resets the anemometer counters to zero and zeroizes all the VALUE array and ANSWER array elements. Following that, the program commences a large loop which is executed twice each minute for the averaging interval requested.

When the program enters this sample loop, the first thing it does is wait until the clock reads 30 seconds then it commences to scan all the sensors and store their values. The anemometer counters are all scanned first and reset to zero followed by all of the ADC inputs and finally the scanlist sensors. Note that the scanlist sensors are multiplexed through the Level Select Unit in "levels" (as explained on page 27). Once all the sensors have been scanned, which takes about 20 seconds, the program loops back and waits for the clock to indicate either 00 or 30 seconds before making another scan. This process continues every thirty seconds until the averaging interval is satisfied.

Upon completion of the averaging interval, the program computes averages of the summed data samples and executes a data output sequence (as per Table XII) to the TTY and the cassette recorder. When the output is complete, the program delays for approximately 20 more seconds (until the start of the next minute) then loops back to the label "CLEAR" where it clears data stores and automatically begins another interval.

E. SYSTEM DEVELOPMENT PROCEDURES

The hardware and software development of MIDAS IIA was aided tremendously by use of the tools and procedures listed


BLANK
PAGE

in the paragraphs below.

1. Intellec 8 Interface

The Intellec 8 Microcomputer Development System (Fig 15) made by Intel Corporation was the backbone of MIDAS IIA development. The Intellec 8 was interconnected to a Datamedia Corporation video terminal (Ref 18 and Fig 15), an Addmaster Corporation high-speed Paper Tape Reader (Ref 19 and Fig 15), a Memodyne cassette recorder (Ref 9), and a TTY as shown in Fig 15.

Since the Intellec 8 uses the same 8008 CPU as the MPS-805 it was ideally suited for software program development. Development was begun by removing all the MPS-805 microprocessor's cards then inserting interface cards in the Input and Output card slots and connecting these cards to Intellec 8 I/O ports. Thus the MIDAS IIA system responded to the commands from the Intellec 8 microprocessor just as it would have with commands from the MPS-805.

Two interface problems were encountered in making the above computer substitution. First of all, the I/O logic levels of the MPS-805 and the Intellec 8 were opposite. This problem was solved by connecting each I/O line to the Intellec 8 through an inverter. Later in the development it became apparent that a better solution to this problem would have been to invert the I/O values in software vice using hardware. The second problem was that the only I/O ports in the Intellec 8 available for use were numbered differently than those used in the MPS-805. This problem was solved by using PL/M "Literals" to identify ports in the PL/M program (see Ref 14 for explanation). These I/O "literals" were easily redefined for one computer

or the other, and by running a new PL/M compile the machine code was completely compatible.

A major benefit of this interface to the Intellec 8 was the ease with which the many program changes could be tested. The change procedure started with a change to the PL/M program. This change was inserted in the program which was stored on disk at the local computer center. A new PL/M compile was then made and a paper tape punched with the new machine code. This paper tape was then brought to the Intellec 8 and read into memory via the high-speed paper tape reader. The new program was then executed on the Intellec 8 and the results noted. A big advantage of using the Intellec 8 to drive the MIDAS IIA system was that program execution could be stopped at any point desired, and the contents of the data storage cells examined to show exactly what was happening. This debugging procedure is not available in the MPS-805.

A side benefit of the Intellec 8 interface was in helping to fault isolate hardware problems by use of a short handwritten machine code program which made a single scan of all sensors and stored the exact results in a certain memory area which was then displayed. By loading and running this program before each new program tape, it was possible to make sure that all the hardware was operating properly, thus insuring that any problems that occurred were a result of software "bugs".

2. PL/M Program Development Procedure

The PL/M program was developed using the "top down structured programming" technique described earlier. References 14 and 15 were used extensively in developing the PL/M code. A timesharing computer system called CP/CMS

(Ref 17) was available from the local computer center via a telephone "modem" interface to a TTY located in the development lab. The CP/CMS system allowed for interactive program changing and recompiling with an immediate indication of any compile errors generated by the changes/additions. Corrections could then be made and another compile attemped. When a successful compile of the program was made, the program was saved on disk at the computer center for future use and the new machine code was outputted to the TTY where a paper tape was punched.

3. PROM Programming Procedures

Use of the Intellec 8 interface during the majority of MIDAS IIA development replaced the need to continually reprogram "PROM" memory chips (for details see page 23) with each change. Reprogramming PROM s is a somewhat tedious procedure that requires removing the PROM s from the ROM card, erasing them under ultraviolet light, reprogramming them, then reinserting them into the ROM card. Although the theoretical cycle life of a PROM is on the order of one million erase cycles, the physical life of a PROM, due to the wear and tear of removal and replacement in the circuit card, is much less than 100 cycles in most cases.

Once the program was fully developed and tested using the Intellec 8 interface, the minor program changes (to the I/O "literals") mentioned previously were made. The program was then compiled and a paper tape of "MPS-805" code was punched. This program was then read into the Intellec 8 memory via the tape reader but not executed. Instead, PROMs were inserted one at the time into a socket on the Intellec 8 face panel and programmed with the MPS-805 program instructions as per Ref 16. The PROMs were installed in the ROM cards then all the MPS-805 circuit

cards were inserted in their proper card slots. At this point the MIDAS IIA system was ready to operate "stand alone" without need for the Intellec 8.

During the course of system development, some problems with PROMs were encountered. Twice a PROM was observed to change its stored program after being heated up to its normal operating temperature of 125° Farenheit. A guide booklet on the proper test and use of PROMs (Ref 5) was very helpful in eliminating these problems. The problem was solved by running all the PROM chips through an acceptance test which involved programming them with all bits set then "cooking" them overnight at 125° F. They were then retested and if any bit had changed the PROM was rejected. Also it was found that the PROMs were programmed more solidly by using the Pro-Log Corporation Model 81 PROM Programmer (Ref 20, available in the development lab) instead of the Intellec 8. The reasons behind this are explained in Ref 5, which is highly recommended to the intended user.

III. MIDAS IIA SYSTEM OPERATION

A. GENERAL

The MIDAS IIA system is designed to operate automatically, once initiated, to repeatedly sample sensors every thirty seconds, compute averages at selected intervals, and output the averages. Overall system operation is closely tied in with the microprocessor's executive program as discussed in Section II D 4.

B. SYSTEM INITIATION

The entire MIDAS IIA system obtains electric power from a single heavy-duty power cable which connects to 110 VAC 60 Hz. This cable feeds the system power supply and numerous electric outlets in the equipment cabinet which in turn power all of the 110 VAC equipments. Once the MIDAS IIA system is connected to a power source, the step-by-step power up/down procedures are performed as listed in Table IX.

The procedures to initialize system operation, after completion of powering up, are given in Table X. As brought out previously, the basic function of the MIDAS IIA program is to periodically sample numerous meteorological sensors and output the average of several samples taken over a selectable time interval. There are presently 34 sensor

inputs of six different types connected to MIDAS IIA as listed in Table II and XI. Four of the six sensor types have a "status" output line which produces a voltage level shift to indicate when the sensor's data lines have valid or invalid information available. The MIDAS IIA program checks the status lines of these four sensor types for the valid data indication before sampling them. A serious problem arises, however, when any one of these four types of sensors is turned off or does not function properly due to a bad probe, broken wire, etc., thus causing a constant invalid indication. If a sensor's status line remained "invalid" then the program could get "hung up" waiting for a valid data indication. To prevent this from occurring, a "SCANLIST" composed of all of these four sensor types was developed as shown in Table VIII. Any position/sensor in the scanlist can be deleted, if determined to be invalid, by use of steps #4 and #5 in Table X.

Table VIII. MIDAS IIA Pre-set Scanlist

1. The pre-set scanlist stored in MIDAS IIA memory (ROM) is enabled when the program is initialized. It stores the values shown below in the positions indicated.

2. The values are encoded to identify the level and type sensor involved as follows:
   VALUE is of the form LS
   where: L = Level (0-4)
   S = Sensor Type (0-4)
   Sensor Types are: 0 = SENSOR DELETED
   1 = HUMIDITY
   2 = H TEMP.
   3 = QUARTZ TEMP.
   4 = BAROMETER

POSITION	VALUE	FUNCTION
01	03	QTZ 0
02	04	BAROM
03	11	HUM 1
04	12	HTMP 1
05	13	QTZ 1
06	21	HUM 2
07	22	HTMP 2
08	23	QTZ 2
09	31	HUM 3
10	32	HTMP 3
11	33	QTZ 3
12	41	HUM 4
13	42	HTMP 4
14	43	QTZ 4
15	FF	END OF LIST

Table IX. MIDAS IIA Power On/Off Procedures

POWER ON PROCEDURES

1. Turn on the master power toggle switch located on the lower center of the Equipment Cabinet (Fig 2 approximately 18 inches above the deck).

2. Turn on the power switch located on the Level Select Unit (Fig 4).

3. Turn on the power switch located above the Memodyne Recorder (Fig 3).

4. Adjust the Digital Clock (Fig 5) to indicate the desired time.

5. Turn on the Digital II Unit (Fig 8) by selecting the "RH" and "°C" lever switches.

6. Turn on the Quartz Thermometer (Fig 7) and set the display interval by twisting the "Display Interval" knob to obtain a sample rate of at least once per second. Set the resolution scale to ".01", the input mode to "T1" and the sensitivity adjustment as required.

7. Insert a cassette in the recorder (Fig 3) and when rewound, engage the tape head by pushing it down. Then press and hold the "Tape Advance" button (located at the top left of cabinet) for 8-10 seconds to advance the tape off of clear leader.

8. Turn on the TTY (Fig 9) to the "System Mode" and activate the paper tape punch.

9. Press the "Reset" button located above the microprocessor (Fig 3) and the TTY will commence outputting the Restart/Initialization sequence as per Table X.

POWER OFF PROCEDURES

1. Switch the TTY to "Local Mode" and punch the space bar a few times to produce a trailer for the paper tape. Next, turn off the paper tape punch and the TTY, then document and save the paper tape for later data reduction. Also save the TTY printout for a master data record.
2. Release the Memodyne Recorder tape head and press the "Rewind" button located above the recorder (Fig 3). Remove the rewound tape then document and save it for later data reduction.
3. Turn off the Quartz Thermometer (Fig 7).
4. Turn off the Digital II Unit (Fig 8).
5. Turn off the power switch above the Memodyne (Fig 3).
6. Turn off the power switch on the Level Select Unit (Fig 4).
7. Turn off the master power toggle switch.

Table X. MIDAS IIA Program Initialization Procedures

1. After "powering up" the system in accordance with the procedures contained in Table IX, begin initialization by pressing the "Reset" button located above the microprocessor (Fig 3).

2. The TTY response is:

   MIDAS IIA RESTART
   INITIALIZE 'Y' OR 'N':

   At this point the operator must respond with a 'Y' (for yes) or 'N' (for no). 'Y' must be chosen when the system is being started after a power interruption or when it is desired to enable the entire pre-set scanlist as per Table VIII. Selecting 'N' will leave the present scanlist unchanged.

3. After initializing, the TTY response is:

   CHANGE SAMPLE LIST 'Y' OR 'N':

   A 'Y' response here allows changes to be made to the sample list (also called scanlist). An 'N' response goes directly to step # 7.

4. If a change request is made, the TTY response is:

   POSITION:

   The operator must enter the two-digit number of one of the scanlist positions shown in Table VIII then press the TTY space bar.

5. Upon selecting a position, for example 08, the TTY adds a response to the current line as follows:

   POSITION: 08 NOW = 23 CHANGE TO:

   The operator may alter the current value (23) assigned to that position using the value coding scheme shown in

Table VIII. For example, to delete the sensor at position 08 (QTZ 2) enter "20" and press the space bar.

6. The above step completes a single sample list change. At this point the program loops back to the query:

   CHANGE SAMPLE LIST 'Y' OR 'N':

   A 'Y' response allows another change cycle as given in steps # 4 and 5.

7. Upon exiting the change sample list sequence (by choosing the 'N' option), the TTY responds with:

   START TIME:

   At this point the operator should enter the desired start time. The time entered is referenced to the MIDAS IIA digital clock (Fig 5) which should be set as required. The entry procedure is to first enter the two digits for the hour and press the space bar, then enter the two digits for the minute and press the space bar again.

   NOTE: The start time must be at least a minute ahead of the current digital clock time to assure proper initialization.

8. Once the desired start time is entered, the TTY responds with:

   AVERAGING INTERVAL (MINUTES):

   The operator should enter the desired averaging interval from 01 to 99 minutes by entering the two-digit value and pressing the space bar. Note that intervals commence at the start time entered, and that a one minute delay between intervals occurs because of the data printout time requirement.

Table XI. MIDAS IIA Memory "Answer" Array

ELEMENT	CONTENTS	ELEMENT	CONTENTS
0	CUP 1	17	ADC 14
1	" 2	18	" 15
2	" 3	19	" 16
3	" 4	20	QTZ0 (SEA TEMP)
4	ADC 1	21	BAROM
5	" 2	22	HUM 1
6	" 3	23	HTEMP 1
7	" 4	24	QTZ 1
8	" 5	25	HUM 2
9	" 6	26	HTEMP 2
10	" 7	27	QTZ 2
11	" 8	28	HUM 3
12	" 9	29	HTEMP 3
13	" 10	30	QTZ 3
14	" 11	31	HUM 4
15	" 12	32	HTEMP 4
16	" 13	33	QTZ 4

Table XII. Sample TTY Printout

1. Following is a sample TTY data printout such as the one that follows the end of a data-gathering and averaging interval. note that the values appear without decimal points thus requiring the following interpretations:

QUARTZ = DD.DD
BAROMETER = DD.DD
H. TEMP. = DDD.D
HUMIDITY = DDD.D
CUPS = R.P.M.
ADC = D.DDD

2. Below the sample output is a key to the printout showing what sensors the various number values represent.

```
  0930
1415   2992
2205   0221   0395   0260   1825   1845   4305   0000
2189   0219   0383   0262   2135   3300   0000   3000
2011   0200   0353   0267   0335   2345   3455   1230
1906   0191   0314   0265   0675   4805   0195   0880


  TIME
QTZ 0  BAROM
QTZ 1  HTMP1  HUM 1  CUP 1  ADC 1  ADC 2  ADC 3  ADC 4
QTZ 2  HTMP2  HUM 2  CUP 2  ADC 5  ADC 6  ADC 7  ADC 8
QTZ 3  HTMP3  HUM 3  CUP 3  ADC 9  ADC10  ADC11  ADC12
QTZ 4  HTMP4  HUM 4  CUP 4  ADC13  ADC14  ADC15  ADC16
```

IV. MIDAS IIA DATA REDUCTION PROCEDURES

A. PAPER TAPE TO PUNCH CARDS

During MIDAS IIA system operation, the TTY paper tape punch is normally activated so that a paper tape record of the printed data is made. This paper tape is formatted such that it can be used to produce a deck of punched hollerith cards through use of the CP/CMS timesharing system (Ref 17). The following procedures presume a basic familiarity with the operation of CP/CMS as described in Ref 17.

First the MIDAS IIA paper tape is checked to be sure it is good by reading a few feet of the beginning in the TTY "local" mode. Next telephone modem contact is established with the computer center and logging into the CP/CMS system is performed as per Ref 17. A new file is initiated in the "edit" mode, thus going into the "edit input" mode. At this point, the paper tape in the TTY tape reader mechanism is prepared and the reader is switched to "AUTO" mode. The paper tape is automatically read into the CP/CMS until it expends itself. When this happens, the reader switch is put to the "STOP" position and the edit input mode is exited by pressing the "XOFF" key twice. Proceeding to the top of the file one makes a global change to the file as follows:

c /(carriage return key)(line feed key)// * *

Figure 15 - MIDAS IIA SYSTEM DEVELOPMENT TOOLS - INTELLEC 8, DATAMEDIA TERMINAL, MEMODYNE RECORDER, AND ADDMASTER PTR

This change deletes all the carriage returns and line feeds which are at the beginning of every record of MIDAS IIA output. Finally, these data are filed and an offline punch of the stored file is executed to obtain a punched card deck.

B. CASSETTE TAPE TO PAPER TAPE

If for some reason the TTY paper tape output of MIDAS IIA data is not available for use in making a punched card deck, as explained in the previous section, it is possible to make another paper tape through use of the cassette tape recording of output data. This paper tape is formatted to be virtually the same as the TTY output version (which is formatted as shown in Table XII) and can be used in its place for making card decks or reading data into the HP-9830A (discussed later). The procedures listed below involve use of the Intellec 8, video terminal, paper tape reader, cassette recorder, and TTY as shown in Fig 15. Also a basic knowledge of Intellec 8 operation, as explained in Ref 16, is presummed.

1. Cassette Tape to Intellec 8 Memory

MIDAS IIA is designed to automatically output data to a cassette tape recorder as well as the TTY, when in normal operation. The cassette thus contains the same data information as the TTY printed copy and paper tape. When a data collection period (e.g. each day) is completed, the cassette is marked and saved as a second data record source. The data may be read from cassette tape into the Intellec 8 memory, when desired, by following the procedures listed in Table XIII.

Table XIII. Procedures for Cassette Tape to Intellec 8 Memory

1. Ensure that the Intellec 8 is on and operating in the Monitor program mode. See the Intellec 8 Operator's Manual (Ref 16) for instructions as necessary.

2. Turn on the Memodyne Model 173 cassette recorder by activating the power switch on the rear of the case. Check the "Mode" switch next to the power switch to be sure it is in the "Intellec 8" position.

3. If the recorder continually rewinds, perform the following steps to correct this problem:
   a. Use the monitor "Substitute" command to put 00 into memory location 2FF0, 10 into 2FF1, 00 into 2FF2, and 01 into 2FF3.
   b. Type in: 'G3000<Return>'
   c. After about seven seconds the cassette recorder should stop rewinding and function properly.

4. Insert a data cassette in the tape deck and rewind it as necessary.
   CAUTION: Always ensure that the recorder tape head is released before pushing "Rewind".

5. Once the tape is rewound, press down on the cassette tape head to engage it.

6. Fill a large block of available memory with a constant value such as 00 by using the monitor "Fill" command. This will be the memory block into which the tape data is read.
   EXAMPLE: F0010,1FFF,00<RETURN>

TABLE XIII (Continued)

Note: Memory locations 2000-20FF should be left vacant, since they are used for the program which controls the output of data to paper tape.

7. Use the monitor "Substitute" command to put into memory locations 2FF0 and 2FF1, the starting memory address to be read into from tape. The upper part of the address goes into 2FF0 and the lower part into 2FF1.
   EXAMPLE: 2FF0 = 00, 2FF1 = 10
   indicates starting address 0010.

8. Substitute into locations 2FF2 and 2FF3 the number (in HEX) of bytes to be read from tape. Again the upper half of the number comes first.
   EXAMPLE: 2FF2 = 1F, 2FF3 = EF
   indicates 1FEF Hex bytes to be read. (This equates to 8,175 decimal bytes which is sufficient to store 11.5 hours of output data taken at 5-minute averaging intervals)

9. Once the four memory locations are set as per the above, the tape "read" operation is performed by typing in 'G3030<Return>'.
   Note: If the data file on the tape is smaller than the number of bytes requested (usually the case), the program will keep advancing the tape and must be stopped by activating the Intellec 8 "RESET" lever switch and releasing the tape head.

10. Use the monitor "Display" command to display the contents of the data block and observe that the tape data has replaced the memory constant value then proceed to the section for outputting the data on paper tape.

2. Intellec 8 Memory to Paper Tape Punch

Once MIDAS IIA output data has been read into an Intellec 8 memory data block from cassette tape, this data block can be outputted to the TTY paper tape punch. The data block is formatted for output by the program listed in Appendix P. Procedures to perform the output are contained in Table XIV.

C. PAPER TAPE TO HP-9830A CALCULATOR

MIDAS IIA TTY data output normally includes a paper tape record of the output as shown in Table XII. This paper tape of the output data may be read into the Hewlett-Packard Model 9830A Calculator (mini-computer, Fig 16) for data reduction and analysis through use of the Addmaster high-speed paper tape reader and a Hewlett/Packard Model 11202A Parallel I/O Interface cable unit. The 11202A Interface is wired for paper tape reader operation as indicated in Appendix F. The physical interfacing procedure is simply to connect the Addmaster reader to the interface cable connector then plug the interface unit into the back of the HP-9830A. Data are then read into the HP-9830A by cueing up a paper tape in the reader and executing a 9830 program, such as the sample contained in Appendix E.

Table XIV. Procedures for Intellec 8 Memory to Paper Tape Punch

1. Use the monitor "Display" function to display the data block to be outputted. Note at the start of the data block, the address number of the first "FF" data byte. Also note at the end of the data block the address of the last "FF" data byte.
   Note: The program to control the output of data to paper tape uses memory locations 2000H thru 20FFH thus if the data block includes these addresses it must be moved through use of the monitor "Move" command before loading the program as given in the next step.

2. Read the Intellec 8 program for control of data output to paper tape (Appendix P) into the Intellec by feeding the program paper tape into the Addmaster paper tape reader and executing the monitor "Read" command. The program loads into memory locations 2000H - 20FFH.

3. Using the "Substitute" command, put into memory locations 2FF0 and 2FF1 the address of the first "FF" data byte in the data block. The upper part of the address goes into 2FF0 and the lower part into 2FF1.

4. Substitute into locations 2FF2 and 2FF3 the address of the last "FF" data byte in the data block. Again the upper part of the address comes first.

TABLE XIV (Continued)

5. Substitute into memory location 2FF4 the number of "levels" of recorded data in the data block (usually 04). The MIDAS IIA system normally records 4 "levels" of sensors but the scanlist can be modified to output from 1 to 4 levels plus the system is designed for eventual expansion to 6 levels.

6. Turn on the ASR-35 Teletype, connected to the Intellec 8, in the "Line" mode and select the "K T" operating mode (K T stands for keyboard copy and paper tape punch).

7. Execute the monitor "Null" command to create a paper tape leader then commence outputting data to the TTY and paper tape punch by typing in at the Intellec 8:

   G2000<Return>

D. CASSETTE TAPE TO HP-9830A CALCULATOR

MIDAS IIA outputs data, such as is shown in Table XII, to a cassette tape recorder as well as the TTY. This cassette tape may be read into the Hewlett-Packard Model 9830A Calculator (mini-computer, Fig 16) for data analysis through use of a Memodyne Model 173 cassette recorder and a Hewlett-Packard Model 11202A Parallel I/O Interface cable unit. The 11202A Interface is wired for cassette tape read operation as indicated in Appendix B. The Memodyne recorder is mounted in a cabinet with an external connector wired as per Appendix C. The Memodyne recorder has been modified for HP-9830A compatibility by the addition of a buffer card as described in Appendix D. A toggle switch on the back of the Memodyne is provided to enable the "HP-9830" mode of operation. The interface unit cable connects to the recorder then the unit itself plugs into the back of the HP-9830A. Preparation to read data is made by first loading a cassette tape in the recorder and rewinding it as necessary. Next the cassette tape head is engaged then the "STOP" key on the HP-9830A is held down until the cassette tape moves off of its clear leader (approximately 6-7 seconds). Finally, tape data are read into the 9830 by executing a program such as the sample contained in Appendix E.

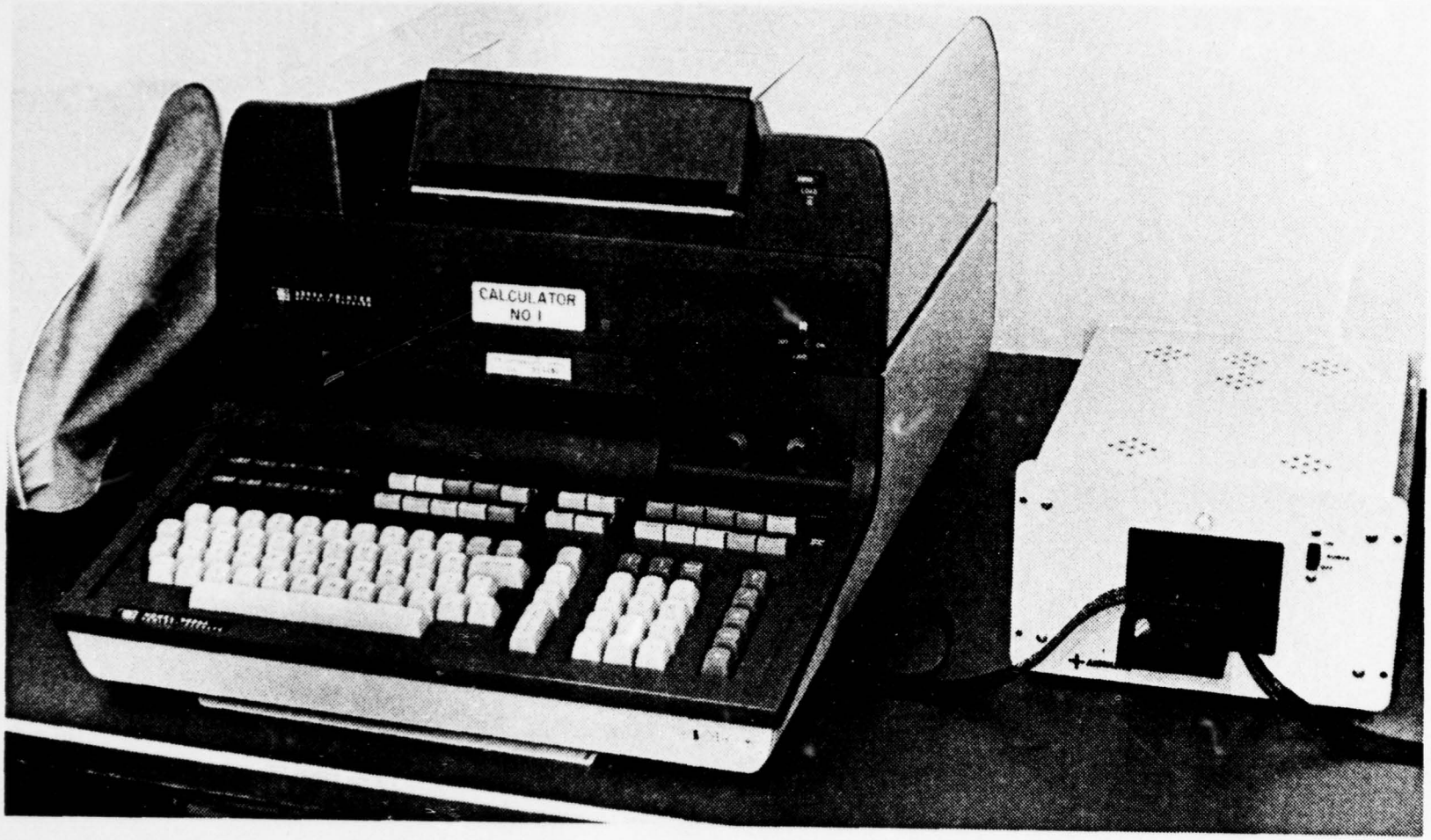


Figure - HP-9830A INTERFACED WITH ADDMASTER PAPER TAPE READER

V. CONCLUSIONS AND RECOMMENDATIONS

The use of a microprocessor as an automated equipment controller for the mean meteorological data system has proven to be quite satisfactory. In fact, the microprocessor is ideally suited to just such tasks because it allows for considerable flexibility and system growth. As an example the MIDAS IIA system presently has four "levels" of sensors, but the hardware and software have been designed and constructed to facilitate future growth (as required) to six levels with minimum effort.

The major area of weakness of the present system is in the input/output terminal device, namely the ASR-33 Teletype. It was originally selected for use because of its low cost and local availability, but its slow data output rate is a serious drawback. For example, the present four-level data output cycle, as shown in Table XII, takes approximately 22 seconds. This lengthy time requires that a minute's delay between data-gathering intervals be inserted in the system's program to allow for output. Based on the above, the need for replacement of the TTY is apparent and is highly recommended. The replacement device should, as a minimum, provide for keyboard input and both a printed output copy and a paper or cassette tape record.

Another area which needs improvement is that of timely data reduction and analysis. It would be very beneficial if the time-averaged data from MIDAS IIA could be reduced and analyzed for validity, content, and trends while still set up and engaged in the data gathering effort aboard ship. This on-board analysis would thus help to insure that

the data collected were sufficient to assure mission success. The solution to this problem is to acquire, for use aboard ship, a programmable, scientific mini-computer such as the Hewlett-Packard 9830A (pictured in Fig 16). The Hewlett-Packard mini-computer is recommended because data reduction procedures for interfacing to the HP-9830A have already been developed (section IV and Appendices A and E) for use on campus. Acquisition of an HP-9830A for on-board computer analysis could also solve the problem of I/O device replacement as discussed above. The HP-9830A could handle the operational input/output requirements of MIDAS IIA with an order of magnitude decrease in time required and still have plenty of time to perform analysis on the developing data base. In fact, by using the HP-9830A as the output device, the data could be stored away in memory for analysis as well as outputted, thus eliminating the need for elaborate data reduction interfaces.

Developments in the microprocessor field have occurred so rapidly that the 8008 CPU, upon which this system is based, is now considered obsolete. The question thus arises as to whether to replace the present "old" microcomputer with a newer model. Based on the following factors, it is recommended that the current microprocessor be retained until a major system revision occurs at some future date. First of all, although the current microprocessor is slow by today's standards, it still has an excess of capability for the MIDAS IIA mean-averaging application. Secondly, the present microprocessor is well supported locally by the Intellec 8 Microcomputer development system (Fig 15) and a computer analyzer for troubleshooting faults. A third factor is the PL/M high-level language support provided by the local computer facility. In view of these considerations, the maintenance of the present configuration is definitely warranted.

APPENDIX A

SAMPLE HP-9830A PROGRAM FOR MIDAS IIA CASSETTE TAPE DATA

```
10    A=RBYTE7
20    IF A # 255 THEN 10
30    REM VARIABLE 'A' NOW CONTAINS FF HEX WHICH MARKS
40    REM THE START OF AN OUTPUT INTERVAL.  THE
50    REM FOLLOWING LINES OF CODE CONVERT AND PACK
60    REM THE DATA VALUES FROM CASSETTE TAPE INTO A
70    REM FORM USABLE BY THE 9830.
80    FOR N=1 TO 2
90    A = RBYTE7
100   B = ROT(A,4)
110   M = 15
120   REM 'M' IS A DATA MASK EQUAL TO 0FH.
130   C = BIAND(B,M)
140   D = BIAND(A,M)
150   IF N=2 THEN 180
160   E = 10 * C + D
170   NEXT N
180   F = 10 * C + D
190   G = 100 * E + F
200   PRINT G
210   REM 'G' CONTAINS A DATA QUANTITY SUCH AS TIME,
220   REM TEMPERATURE, ETC. WHICH COULD AT THIS
230   REM POINT BE STORED AWAY IN AN APPROPRIATE
240   REM ARRAY FOR A DATA ANALYSIS PROGRAM
250   GOTO 10
```

APPENDIX B

HP-9830A I/O INTERFACE # 11202A WIRING FOR CASSETTE TAPE OPERATION

1. Connect +5V to Memodyne Pin #24 (FWD/REV*) to insure forward operation by strapping it to pin #25 (+5 Volts).

2. * = Logic zero value indicated or required.

HP #11202A PARALLEL I/O INTERFACE CABLE		MEMODYNE CASSETTE RECORDER CONNECTOR AS PER APPENDIX 00	
FUNCTION	WIRE COLOR	PIN #	FUNCTION
INPUT 1	BLACK	14	OUTPUT 1
2	BROWN	15	2
3	RED	16	3
4	ORANGE	17	4
5	YELLOW	18	5
6	GREEN	19	6
7	BLUE	20	7
8	VIOLET	21	8
CTL*	WH/GRAY	11	START*
FLG*	GRAY	23	TAPE SYNC*
STP*	WH/BLK/RED	10	LOAD FWD*
GRND	WH/BLK/ORNG	1,13	GND,READ*
		24,25	STRAP TOGETHER

APPENDIX C

MEMODYNE CASSETTE RECORDER INPUT/OUTPUT CONNECTIONS

1. The following pin numbers refer to the 25 pin CANNON female connector located on the side of the case of the MEMODYNE recorder Model #173.

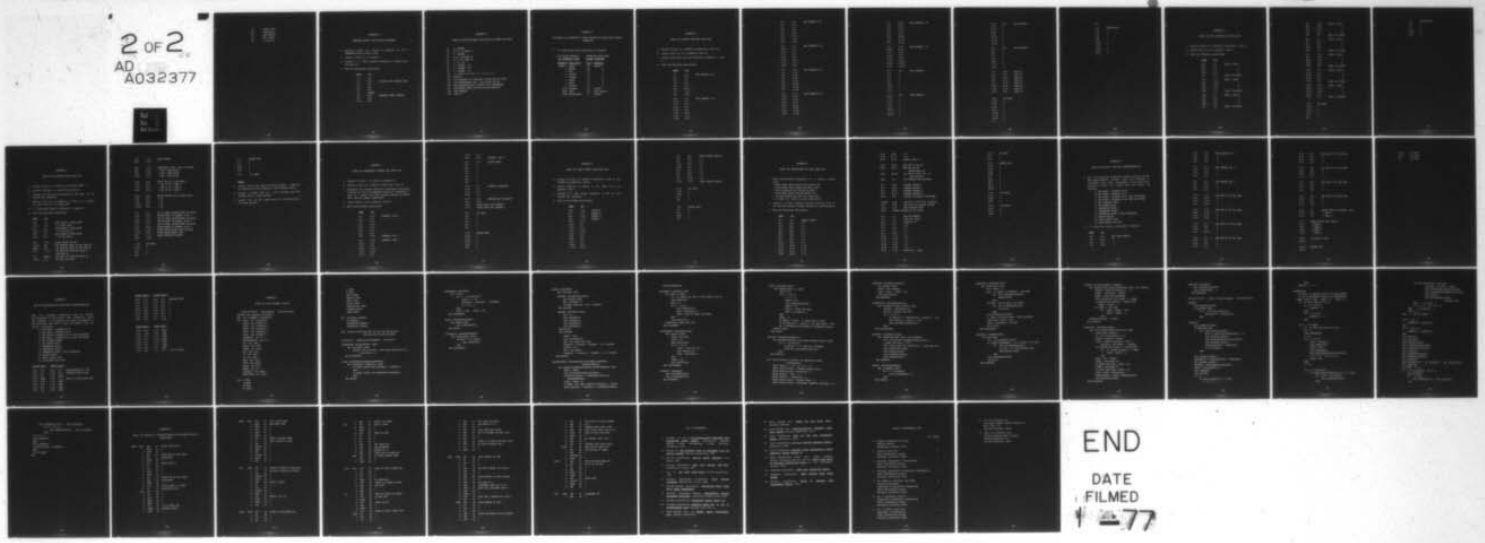
2. ** = Logic zero indicated or required.

PIN #	FUNCTION
1	GROUND
2	INPUT BIT 1
3	INPUT BIT 2
4	INPUT BIT 3
5	INPUT BIT 4
6	INPUT BIT 5
7	INPUT BIT 6
8	INPUT BIT 7
9	INPUT BIT 8
10	LOAD FORWARD**
11	START**
12	START
13	READ** / WRITE
14	OUTPUT BIT 1
15	OUTPUT BIT 2
16	OUTPUT BIT 3
17	OUTPUT BIT 4
18	OUTPUT BIT 5
19	OUTPUT BIT 6
20	OUTPUT BIT 7

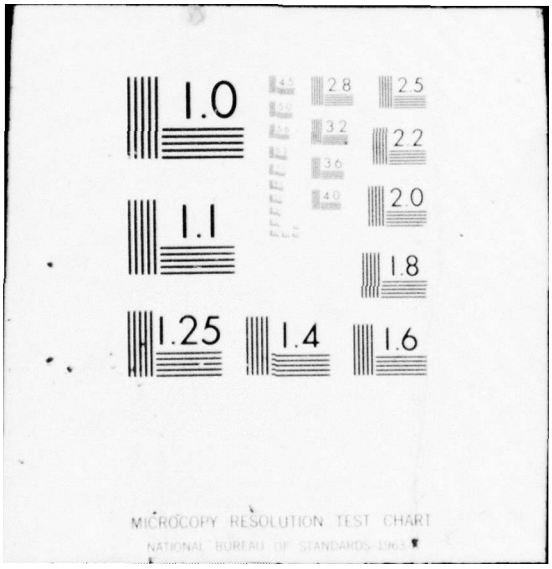
AD-A032 377 NAVAL POSTGRADUATE SCHOOL MONTEREY CALIF F/G 9/2
A MICROPROGRAMMABLE DATA ACQUISITION AND CONTROL SYSTEM (MIDAS --ETC(U)
SEP 76 J K PLUNKETT

UNCLASSIFIED NL

2 OF 2
AD
A032377



END
DATE
FILMED
1-77


1.0
1.1
1.25
1.4
1.6
2.8
2.5
3.2
2.2
3.6
2.0
4.0
1.8
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963

21	OUTPUT BIT 8
22	(BOT/EOT) **
23	TAPE SYNC**
24	FWD / REV**
25	+5 VOLT DC

APPENDIX D

MEMODYNE BUFFER FOR HP-9830A INTERFACE

1. Install a 14-pin I.C. socket in position #1 of a MEMODYNE utility circuit card.
2. Insert a 74122 I.C. in socket.
3. Connect a .1 mfd. Capacitor between I.C. socket pins #11 and #13.
4. Make the following connections:

FROM	TO	
1-1	VCC	
1-2	E-21	HP-9830A CTL COMMAND LINE
1-3	VCC	
1-4	VCC	
1-5	VCC	
1-7	GROUND	
1-8	E-12	MEMODYNE START COMMAND
1-9	VCC	
1-14	VCC	

APPENDIX E

SAMPLE HP-9830A PROGRAM FOR MIDAS IIA PAPER TAPE DATA

```
10    A = RBYTE6
20    IF A # 13 THEN 10
30    A = RBYTE6
40    IF A < 48 THEN 30
50    IF A > 57 THEN 10
60    A = A - 48
70    B = RBYTE6 - 48
80    C = RBYTE6 - 48
90    D = RBYTE6 - 48
100   E = 1000 * A + 100 * B + 10 * C + D
110   PRINT E
120   REM VARIABLE 'E' CONTAINS A VALUE SUCH AS TIME,
130   REM TEMPERATURE, ETC.  THIS VALUE CAN BE
140   REM PROGRAMATICALLY STORED IN AN APPROPRIATE
150   REM MEMORY ARRAY FOR USE IN DATA ANALYSIS
160   REM PROGRAMS
170   GOTO 30
```

APPENDIX F

HP-9830A I/O INTERFACE # 11202A WIRING FOR PAPER TAPE READER OPERATION

1. * = Logic zero value indicated or required.

HP #11202A PARALLEL I/O INTERFACE CABLE		ADDMASTER PAPER TAPE READER CONNECTOR	
FUNCTION	WIRE COLOR	PIN #	FUNCTION
INPUT 1	BLACK	09	OUTPUT 1
2	BROWN	08	2
3	RED	07	3
4	ORANGE	06	4
5	YELLOW	13	5
6	GREEN	12	6
7	BLUE	11	7
8	VIOLET	10	8
CTL*	WH/GRAY	01	CLOCK*
FLG*	GRAY	03	DATA AVAIL.*
GRND	WH/BLK/ORNG	17	GROUND

APPENDIX G

MIDAS IIA DIGITAL MUX CARD WIRE LIST

1. Install 16-pin I.C. sockets in positions 1 thru 12.
2. Insert 74151 I.C.'s in sockets 5 thru 12.
3. Digital MUX inputs will be connected to sockets 1 thru 4.
4. Make the following connections:

FROM	TO	
1-1	5-4	MUX CHANNEL 0 IN
1-2	6-4	
1-3	7-4	
1-4	8-4	
1-5	9-4	
1-6	10-4	
1-7	11-4	
1-8	12-4	
1-9	5-3	MUX CHANNEL 1 IN
1-10	6-3	
1-11	7-3	
1-12	8-3	
1-13	9-3	
1-14	10-3	
1-15	11-3	
1-16	12-3	

2-1	5-2	MUX CHANNEL 2 IN
2-2	6-2	
2-3	7-2	
2-4	8-2	
2-5	9-2	
2-6	10-2	
2-7	11-2	
2-8	12-2	
2-9	5-1	MUX CHANNEL 3 IN
2-10	6-1	
2-11	7-1	
2-12	8-1	
2-13	9-1	
2-14	10-1	
2-15	11-1	
2-16	12-1	
3-1	5-15	MUX CHANNEL 4 IN
3-2	6-15	
3-3	7-15	
3-4	8-15	
3-5	9-15	
3-6	10-15	
3-7	11-15	
3-8	12-15	
3-9	5-14	MUX CHANNEL 5 IN
3-10	6-14	
3-11	7-14	
3-12	8-14	
3-13	9-14	
3-14	10-14	
3-15	11-14	
3-16	12-14	

4-1	5-13	MUX CHANNEL 6 IN
4-2	6-13	
4-3	7-13	
4-4	8-13	
4-5	9-13	
4-6	10-13	
4-7	11-13	
4-8	12-13	
4-9	5-12	MUX CHANNEL 7 IN
4-10	6-12	
4-11	7-12	
4-12	8-12	
4-13	9-12	
4-14	10-12	
4-15	11-12	
4-16	12-12	
5-7	E-B	CARD SELECT*
6-7	"	
7-7	"	
8-7	"	
9-7	"	
10-7	"	
11-7	"	
12-7	"	
5-11	E-E	DATA SELECT 1
6-11	"	
7-11	"	
8-11	"	
9-11	"	
10-11	"	
11-11	"	
12-11	"	

5-10	E-D	DATA SELECT 2
6-10	"	
7-10	"	
8-10	"	
9-10	"	
10-10	"	
11-10	"	
12-10	"	
5-9	E-C	DATA SELECT 4
6-9	"	
7-9	"	
8-9	"	
9-9	"	
10-9	"	
11-9	"	
12-9	"	
5-6	E-10	DATA 8*
6-6	E-11	DATA 7*
7-6	E-12	DATA 6*
8-6	E-13	DATA 5*
9-6	E-14	DATA 4*
10-6	E-15	DATA 3*
11-6	E-16	DATA 2*
12-6	E-17	DATA 1*
5-16	VCC BUSS	
6-16	"	
7-16	"	
8-16	"	
9-16	"	
10-16	"	
11-16	"	
12-16	"	
E-1	"	

E-A	"
5-8	GROUND BUSS
6-8	"
7-8	"
8-8	"
9-8	"
10-8	"
11-8	"
12-8	"
E-22	"
E-Z	"

APPENDIX H

MIDAS IIA MUX COMBINER CARD WIRE LIST

1. Install 14-pin I.C. sockets in positions 1 thru 4.
2. Insert 7420 I.C.'s in sockets 1 thru 4.
3. Make the following connections:

FROM	TO	
E-B	1-1	DATA 8, MUX 0
E-C	1-2	1
E-D	1-4	2
E-E	1-5	3
E-F	1-6	DATA 8 TO MICRO
E-H	1-13	DATA 7, MUX 0
E-J	1-12	1
E-K	1-10	2
E-L	1-9	3
E-M	1-8	DATA 7 TO MICRO
E-N	2-1	DATA 6, MUX 0
E-P	2-2	1
E-R	2-4	2
E-S	2-5	3
E-T	2-6	DATA 6 TO MICRO

E-U	2-13	DATA 5, MUX 0
E-V	2-12	1
E-W	2-10	2
E-X	2-9	3
E-Y	2-8	DATA 5 TO MICRO
E-2	3-1	DATA 4, MUX 0
E-3	3-2	1
E-4	3-4	2
E-5	3-5	3
E-6	3-6	DATA 4 TO MICRO
E-7	3-13	DATA 3, MUX 0
E-8	3-12	1
E-9	3-10	2
E-10	3-9	3
E-11	3-8	DATA 3 TO MICRO
E-12	4-1	DATA 2, MUX 0
E-13	4-2	1
E-14	4-4	2
E-15	4-5	3
E-16	4-6	DATA 2 TO MICRO
E-17	4-13	DATA 1, MUX 0
E-18	4-12	1
E-19	4-10	2
E-20	4-9	3
E-21	4-8	DATA 1 TO MICRO

1-14	VCC BUSS
2-14	"
3-14	"
4-14	"
E-1	"
E-A	"

1-7	GROUND BUSS
2-7	"
3-7	"
4-7	"
E-22	"
E-Z	"

APPENDIX I

MIDAS IIA TTY/BUFFER CARD WIRE LIST

1. Install 14-pin I.C. sockets in positions 1 and 2.
2. Install a 16-pin I.C. socket in position 4.
3. Connect 2.7K Ohm pull-up resistors to VCC Buss at all blocked (•) terminals.
4. Insert a 7417 I.C. in socket #1, a 7416 I.C. in socket #2, and a 7445 I.C. in socket #4.
5. * = Logic zero value is indicated or required.
6. Make the following connections:

FROM	TO	
E-D	1-5	DATA SELECT 1 FROM MICRO
1-6•	E-4	" TO MUX CARDS
E-C	1-3	DATA SELECT 2 FROM MICRO
1-4•	E-3	" TO MUX CARDS
E-B	1-1	DATA SELECT 4 FROM MICRO
1-2•	E-2	" TO MUX CARDS
E-E	1-13	MICRO OUTPUT TO TTY
1-12•	E-5	TTY RECEIVE INPUT TO TTY TERM #6
NOTE 1	E-H	TTY RECEIVE INPUT TO TTY TERM #7
E-6	1-11•	TTY XMIT OUT FROM TTY TERM #3 (SEE NOTE 2)
E-7	NOTE 3	TTY XMIT OUT FROM TTY TERM #4
1-10•	E-J	TTY XMIT TO MICRO

E-K	1-9	SPARE DRIVER
E-8	1-8■	"
E-L	4-12	CARD/FUNCT. ADDR. SEL 8 FM MICRO
E-9	4-13	" SEL 4 FROM MICRO
E-M	4-14	" SEL 2 FROM MICRO
E-10	4-15	" SEL 1 FROM MICRO
4-1■	E-N	MICRO SEL 0* TO MUX CARD 0
4-2■	E-11	" SEL 1* TO " CARD 1
4-3■	E-P	" SEL 2* TO " CARD 2
4-4■	E-12	" SEL 3* TO " CARD 3
4-6■	E-13	SPARE CONTROL SEL 5* FROM MICRO
4-7■	E-S	" 6*
4-9■	E-14	" 7*
4-10■	E-T	" 8*
4-11■	E-15	" 9*
4-5	2-1	SEL 4* ANEM. RESET*/COUNT FM MICRO
2-2■	E-16	RESET/COUNT* TO ANEMOM. CARD 1
4-5	2-3	SEL 4* ANEM. RESET*/COUNT FM MICRO
2-4■	E-17	RESET/COUNT* TO ANEMOM. CARD 2
4-5	2-5	SEL 4* ANEM. RESET*/COUNT FM MICRO
2-6■	E-18	RESET/COUNT* TO ANEMOM. CARD 3
4-5	2-13	SPARE RESET*/COUNT LINE FM MICRO
2-12■	E-19	SPARE RESET/COUNT* LINE
E-Y	2-11	SPARE INVERTING DRIVER
E-20	2-10■	"
1-14	VCC BUSS	
2-14	"	
4-16	"	
E-1	"	
E-A	"	

1-7	GROUND BUSS
2-7	"
4-8	"
E-22	"
E-Z	"
E-W	-10 VOLTS

NOTES:

1. Connect E-H to VCC thru a 220 Ohm resistor. Edge pin E-H connects to TTY terminal #7 (TTY receive input).
2. Connect a diode from pin 1-11 to ground such that current flow is toward ground.
3. Connect E-7 to E-W (which goes to -10 volts) thru a 2.7K Ohm resistor.

APPENDIX J

MIDAS IIA ANEMOMETER COUNTER CARD WIRE LIST

1. Install a 16-pin I.C. socket in position 2.
2. Install 14-pin I.C. sockets in positions 5 thru 8.
3. Establish TTL voltage levels by installing level-shift circuits for two cup anemometer inputs in positions 9 thru 12. Bypass the level shifted outputs to ground with 0.05 MFD ceramic capacitors.
4. Insert 7493 I.C.'s in sockets 5 thru 8.
5. Make the following connections:

FROM	TO	
2-1	5-11	COUNTER 1 BIT 8
2-2	5-8	
2-3	5-9	
2-4	5-12	
2-5	6-11	
2-6	6-8	
2-7	6-9	
2-8	6-12	COUNTER 1 BIT 1
2-9	7-11	COUNTER 2 BIT 8
2-10	7-8	
2-11	7-9	
2-12	7-12	
2-13	8-11	
2-14	8-8	

2-15	8-9	
2-16	8-12	COUNTER 2 BIT 1
5-2	E-3	RESET/COUNT*
5-3	"	
6-2	"	
6-3	"	
7-2	"	
7-3	"	
8-2	"	
8-3	"	
5-12	5-1	INTERNAL PROPAGATE
6-12	6-1	
7-12	7-1	
8-12	8-1	
6-11	5-14	HEXADECIMAL PROPAGATE
8-11	7-14	
6-14	LEVEL SHIFT OUT COUNTER 1	
8-14	LEVEL SHIFT OUT COUNTER 2	
5-5	VCC BUSS	
6-5	"	
7-5	"	
8-5	"	
E-1	"	
E-A	"	
5-10	GROUND BUSS	
6-10	"	
7-10	"	
8-10	"	
E-22	"	
E-Z	"	

APPENDIX K

MIDAS IIA LEVEL SELECT CARD WIRE LIST

1. Install a 16 pin I.C. socket in position 1, and 14 pin sockets in positions 3 and 4.
2. Insert a 7445 I.C. in socket 1, and 7404 I.C.'s in sockets 3 and 4.
3. Install 2.7 K Ohm pullup resistors to VCC at all starred (*) terminals.
4. Make the following connections:

FROM	TO	
E-B	1-12	SELECT 8
E-C	1-13	SELECT 4
E-D	1-14	SELECT 2
E-E	1-15	SELECT 1
1-1*	3-1	
1-2*	3-3	
1-3*	3-5	
1-4*	3-13	
1-5*	3-11	
1-6*	3-9	
1-7*	4-1	
1-9*	4-3	
1-10*	4-5	
1-11*	4-13	

3-2	E-2	LEVEL SELECT LEVEL 0
3-4	E-3	" 1
3-6	E-4	" 2
3-12	E-5	" 3
3-10	E-6	" 4
3-8	E-7	" 5
4-2	E-8	" 6
4-4	E-9	" 7
4-6	E-10	" 8
4-12	E-11	LEVEL SELECT LEVEL 9

1-16	VCC BUSS
3-14	"
4-14	"
E-1	"
E-A	"

1-8	GROUND BUSS
3-7	"
4-7	"
E-22	"
E-Z	"

APPENDIX L

MIDAS IIA MULTIPLEXED ADC CARD WIRE LIST

1. Mount the following components on a utility circuit card:
   - Datel Model MM-16 Analog Multiplexer (M)
   - Datel Model SHM-4 Sample and Hold (SH)
   - 10K Ohm Trimpot for Voltage Divider.
   - Datel Model ADCE-12B2 ADC Module (A)
   - 100 Ohm Trimpot for ADC Full Scale Adj.
   - A 16-Pin I.C. Socket in card position #4.

2. Connect a .01 mfd. Capacitor between A-23 and A-24 to set the ADC internal trigger spacing at 8 milliseconds.

3. Make the following connections:

FROM	TO	
E-2	M-1	ANALOG INPUT 1
E-3	M-2	" 2
E-4	M-3	" 3
E-5	M-4	" 4
E-6	M-5	" 5
E-7	M-6	" 6
E-8	M-7	" 7
E-9	M-8	" 8
E-10	M-9	" 9
E-11	M-10	" 10
E-12	M-11	" 11
E-13	M-12	" 12
E-14	M-13	" 13

E-15	M-14	" 14
E-16	M-15	" 15
E-17	M-16	ANALOG INPUT 16
M-27	M-26	MUX OUT TO AMP IN
M-25	SH-13	MUX AMP OUT TO SAMPLE/HOLD IN
SH-4	VD-IN	S/H OUT TO VOLT. DIV. IN (10K OHM POT)
SH-9	E-F	S/H COMMAND FROM MICROPRO.
E-B	M-20	CHANNEL ADDRESS 1
E-C	M-21	CHANNEL ADRESS 2
E-D	M-22	CHANNEL ADDRESS 4.
E-E	M-23	CHANNEL ADDRESS 8
M-24	VCC	INHIBIT DE-SELECT (PERMANENT)
VD-OUT	A-28	VOLT.DIV. OUT TO ADC INPUT(HI)
A-27	A-26	ADC IN(LO) TO ANALOG GROUND
A-22	A-21	SELF-TRIGGER MODE
A-25	1000HM TRIMPOT TO +15 VOLTS	
A-1	4-1	END-OF-CONVERT
A-2	4-5	SIGN BIT (MSB)
A-3	4-6	DATA BIT 11
A-6	4-7	" 10
A-7	4-8	" 9
A-8	4-9	" 8
A-9	4-10	" 7
A-10	4-11	" 6
A-11	4-12	" 5
A-12	4-13	" 4
A-13	4-14	" 3
A-14	4-15	" 2
A-15	4-16	DATA BIT 1 (LSB)

A-17	VCC BUSS
E-1	"
E-A	"
M-19	GROUND BUSS
A-16	"
A-20	"
A-26	"
E-22	"
E-Z	"
SH-3	"
SH-7	"
SH-8	"
SH-12	"
M-17	+15V BUSS
A-18	"
E-20	"
SH-5	"
M-18	-15V BUSS
A-19	"
E-21	"
SH-6	"

APPENDIX M

MIDAS IIA UTILITY CARD CAGE INTERCONNECTIONS

1. Make the following connections between utility circuit card edge-connector sockets, which are numbered to correspond with their arrangement in the card cage, from left to right (as viewed from the front), as follows:

    1. MUX BOARD 1 CHANNELS 00-07
    2. MUX BOARD 2 CHANNELS 10-17
    3. MUX BOARD 3 CHANNELS 20-27 (NOT INSTALLED)
    4. MUX BOARD 4 CHANNELS 30-37 (NOT YET WIRED)
    5. MUX COMBINER BOARD
    6. TTY BUFFER BOARD
    7. ANEMOMETER BOARD 1
    8. ANEMOMETER BOARD 2
    9. ANEMOMETER BOARD 3 (NOT INSTALLED)
    10. QUARTZ BUFFER
    11. LEVEL SELECT CARD
    12. MULTIPLEXED ADC CARD

2. * = Logic zero value is indicated or required.

FROM	TO	
1-B	6-N	MUX CARD SELECT*
2-B	6-11	"
3-B	6-P	"
4-B	6-12	"

1-E	6-4	MUX CHANNEL SEL 1
2-E	6-4	"
3-E	6-4	"
4-E	6-4	"
1-D	6-3	MUX CHANNEL SEL 2
2-D	6-3	"
3-D	6-3	"
4-D	6-3	"
1-C	6-2	MUX CHANNEL SEL 4
2-C	6-2	"
3-C	6-2	"
4-C	6-2	"
1-17	5-17	MUX DATA 1* TO MUX COMB.
2-17	5-18	"
3-17	5-19	"
4-17	5-20	"
1-16	5-12	MUX DATA 2* TO MUX COMB.
2-16	5-13	"
3-16	5-14	"
4-16	5-15	"
1-15	5-7	MUX DATA 3* TO MUX COMB.
2-15	5-8	"
3-15	5-9	"
4-15	5-10	"
1-14	5-2	MUX DATA 4* TO MUX COMB.
2-14	5-3	"
3-14	5-4	"
4-14	5-5	"

1-13	5-U	MUX DATA 5* TO MUX COMB.
2-13	5-V	"
3-13	5-W	"
4-13	5-X	"
1-12	5-N	MUX DATA 6* TO MUX COMB.
2-12	5-P	"
3-12	5-R	"
4-12	5-S	"
1-11	5-H	MUX DATA 7* TO MUX COMB.
2-11	5-J	"
3-11	5-K	"
4-11	5-L	"
1-10	5-B	MUX DATA 8* TO MUX COMB.
2-10	5-C	"
3-10	5-D	"
4-10	5-E	"
6-16	7-3	RESET/COUNT* TO ANEMOM. CARD 1
6-17	8-3	" CARD 2
6-18	9-3	" CARD 3
12-2	LEVEL SELECT UNIT LEVEL 0	
12-3	" LEVEL 1	
12-4	" LEVEL 2	
12-5	" LEVEL 3	
12-6	" LEVEL 4	
ALL-1	+5 VOLT VCC BUSS	
ALL-A	"	
ALL-22	GROUND BUSS	
ALL-Z	"	

12-20	+15 VOLTS
12-21	-15 VOLTS
6-W	-10 VOLTS

APPENDIX N

MPS-805 MICROPROCESSOR CARD CAGE INTERCONNECTIONS

1. Make the following connections from the MPS-805 Microprocessor to the utility cage connectors, which are numbered to correspond with their arrangement in the card cage, from left to right (as viewed from the front), as follows:

    1. MUX BOARD 1 CHANNELS 00-07
    2. MUX BOARD 2 CHANNELS 10-17
    3. MUX BOARD 3 CHANNELS 20-27 (NOT INSTALLED)
    4. MUX BOARD 4 CHANNELS 30-37 (NOT YET WIRED)
    5. MUX COMBINER BOARD
    6. TTY BUFFER BOARD
    7. ANEMOMETER BOARD 1
    8. ANEMOMETER BOARD 2
    9. ANEMOMETER BOARD 3 (NOT INSTALLED)
    10. QUARTZ BUFFER
    11. LEVEL SELECT CARD
    12. MULTIPLEXED ADC CARD

OUTPUT PORT 0		OUTPUT PORT 2		
O-56	6-D	O-22	6-E	MICROPROCESSOR TO TTY
O-54	6-C	O-20	M-12	MEMODYNE START PULSE
O-52	6-B	O-18	OPEN	
O-50	OPEN	O-16	12-F	MICRO TO SAMPLE/HOLD CMD
O-48	6-10	O-14	OPEN	
O-46	6-M	O-12	OPEN	
O-44	6-9	O-10	OPEN	
O-42	6-L	O-8	OPEN	

OUTPUT PORT 1		OUTPUT PORT 3		
O-55	11-E	O-21	M-M	MEMODYNE DATA
O-53	11-D	O-19	M-N	"
O-51	11-C	O-17	M-U	"
O-49	11-B	O-15	M-V	"
O-47	12-B	O-13	M-W	"
O-45	12-C	O-11	M-X	"
O-43	12-D	O-9	M-Y	"
O-41	12-E	O-7	M-19	"

INPUT PORT 0		INPUT PORT 1		
I-55	5-21	I-21	OPEN	
I-53	5-16	I-19	OPEN	
I-51	5-11	I-17	OPEN	
I-49	5-6	I-15	OPEN	
I-47	5-Y	I-13	OPEN	
I-45	5-T	I-11	OPEN	
I-43	5-M	I-9	OPEN	
I-41	5-F	I-7	6-J	TTY TO MICRO.

APPENDIX O

MIDAS IIA PL/M PROGRAM LISTING

```
   *************   DECLARATIONS   ***************
DECLARE DEC LITERALLY 'DECLARE';
DEC  LIT  LITERALLY 'LITERALLY';
DEC  OUT00  LIT 'OUTPUT(0)',
     OUT01  LIT 'OUTPUT(1)',
     OUT02  LIT 'OUTPUT(2)',
     OUT03  LIT 'OUTPUT(3)',
     INP00  LIT 'INPUT(0)',
     INP01  LIT 'INPUT(1)',
     MAX$LEVEL  LIT '4',
     MAX$ADC$VOLTS  LIT '5',
     ADC  LIT '10H',
     BAROM  LIT '02H',
     HTEMP  LIT '14H',
     HUM  LIT '16H',
     CR  LIT '0DH',
     LF  LIT '0AH',
     XOFF  LIT '13H',
     TRUE  LIT '0FFH',
     FALSE  LIT '00',
     DUMMY  LIT '0',
     ZEROSECS  LIT '0FFH',
     THIRTYSECS  LIT '0FH';

DEC  I BYTE,
     J BYTE,
     K BYTE,
     K1 BYTE,
```

```
     L BYTE,
     M BYTE,
     POSIT BYTE,
     ENDLIST BYTE,
     LEVEL BYTE,
     HMS(6) BYTE,
     SCANLIST(21) BYTE,
     INTERVAL BYTE,
     NSAMP BYTE;

DEC  VALUE(80) ADDRESS,
     HM ADDRESS,
     START$TIME ADDRESS,
     ANSWER(40) ADDRESS;

DEC  INITLIST DATA(03H,04H,11H,12H,13H,21H,22H,23H,
                   31H,32H,33H,41H,42H,43H,0FFH;

*********   SUBROUTINE PROCEDURES   **********

BCD$TO$HEX: PROCEDURE(BCD) BYTE;
     DEC (BCD,XPAD) BYTE;
          RETURN( (BCD AND 0FH) + SHL((XPAD:=SHR(BCD,4)),1)
               + SHL(XPAD,3) );
   END BCD$TO$HEX;

PULSE: PROCEDURE(PULS$MASK,MATCH$VAL);
     DEC (PULS$MASK, MATCH$VAL) BYTE;
          DO WHILE (INP00 AND PULS$MASK) = MATCHVAL;
          END;
          DO WHILE (INP00 AND PULS$MASK)<>MATCH$VAL;
          END;
   END PULSE;
```

```
ANEMOMETER: PROCEDURE;
     DEC I BYTE;
          DO I = 1 TO MAX$LEVEL;
               OUT00 = I + 3;
               VALUE(K1) = VALUE(K1) + (2*INP00);
               K1 = K1 + 2;
          END;
          OUT00 = 40H;   OUT00 = 00;
   END ANEMOMETER;

DELAY: PROCEDURE(AMOUNT);
     DEC AMOUNT BYTE;
          CALL TIME(AMOUNT);
   END DELAY;

LEVL$DELAY: PROCEDURE(LNGTH);
     DEC (LNGTH,XPAD) BYTE;
          DO XPAD = 0 TO LNGTH;
               CALL DELAY(255);
          END;
   END LEVL$DELAY;
```

```
QUARTZ: PROCEDURE;
     DEC (VAL,NUM) BYTE;

     SHIFTER: PROCEDURE(PLUSVAL);
          DEC PLUSVAL BYTE;
          NUM = ROR(NUM,1);
          IF CARRY THEN VAL = VAL + PLUSVAL;
      END SHIFTER;

     CONVERT: PROCEDURE BYTE;
          VAL = 0;
          CALL SHIFTER(1);
          CALL SHIFTER(2);
          CALL SHIFTER(2);
          CALL SHIFTER(4);
      RETURN VAL;
      END CONVERT;

          OUT00 = 12H;
          CALL PULSE(80H,0);
          NUM = NOT INP00 AND 7FH;
          VALUE(K) = VALUE(K) + CONVERT + 10 * CONVERT;
          OUT00 = 13H;
          NUM = NOT INP00;
          VALUE(K1) = VALUE(K1) + CONVERT + 10 * CONVERT;
   END QUARTZ;

MULTI$SENSOR: PROCEDURE(NAME,PULSS$MASK,MACH$VAL,
                        UPPERWORD$MASK);
     DEC (NAME,PULSS$MASK,MACH$VAL,UPPERWORD$MASK) BYTE;
          OUT00 = NAME;
          CALL PULSE(PULSS$MASK,MACH$VAL);
          VALUE(K)=VALUE(K) + BCD$TO$HEX(INP00 AND
             UPPERWORD$MASK);
          OUT00 = NAME + 1;
          IF NAME = ADC THEN VALUE(K1)=VALUE(K1) + INP00;
          ELSE VALUE(K1) = VALUE(K1) + BCD$TO$HEX(INP00);
```

```
   END MULTI$SENSOR;

TYCHAR$IN: PROCEDURE BYTE;
     DEC (XPAD,I) BYTE;
          T1:  IF (INP01 AND 80H) <> 80H THEN GO TO T1;
          CALL DELAY(64);
          XPAD = 0;
          DO I = 1 TO 8;
               CALL DELAY(122);
               XPAD = ROR(XPAD,1);
               XPAD = XPAD OR (INP01 AND 80H);
          END;
          CALL DELAY(122);
     RETURN((NOT XPAD) AND 7FH);
   END TYCHAR$IN;

TTYCHAR$OUT: PROCEDURE(CHAR);
     DEC (CHAR,I) BYTE;
          CHAR = NOT CHAR;
          OUT02 = 01;
          CALL DELAY(122);
          DO I = 1 TO 8;
               OUT02 = CHAR AND 01H;
               CALL DELAY(122);
               CHAR = ROR(CHAR,1);
          END;
          OUT02 = 00;
          CALL DELAY(122);
   END TTYCHAR$OUT;

TTY$CRLF: PROCEDURE;
     CALL TTYCHAR$OUT(CR);
     CALL TTYCHAR$OUT(LF);
   END TTY$CRLF;
```

```
CLOCK: PROCEDURE BYTE;
     DEC (READY,XPAD,I,J) BYTE;
          XPAD = 01;
          DO I = 1 TO 6;
               J = 6 - I;
               OUT00 = 00;
               CALL PULSE(XPAD,XPAD);
               OUT00 = 01;
               HMS(J) = INP00 AND 0FH;
               XPAD = ROL(XPAD,1);
          END;
          READY = 0;
          IF HMS(4) + HMS(5) = 0 THEN READY = 0FFH;
          IF SHL(HMS(4),4) + HMS(5) = 30H THEN READY = 0FH;
          HM = 1000H*HMS(0)+100H*HMS(1)+10H*HMS(2)+HMS(3);
     RETURN READY;
   END CLOCK;

MESSAGE: PROCEDURE(MSG$ADDR);
     DEC MSG$ADDR ADDRESS, MSG BASED MSG$ADDR BYTE,I BYTE;
          DO I = 1 TO MSG(0);
               IF MSG(I) = '*' THEN CALL TTY$CRLF;
               ELSE CALL TTYCHAR$OUT(MSG(I));
          END;
   END MESSAGE;

DEC  MSG00 DATA(34,'***MIDAS IIA RESTART**',XOFF,
                ' INITIALIZE'),
     MSG01 DATA(13,' ''Y'' OR ''N'': '),
     MSG02 DATA(22,XOFF,' **CHANGE SAMPLE LIST'),
     MSG03 DATA(13,XOFF,' *POSITION: '),
     MSG04 DATA(8,'  NOW = '),
     MSG05 DATA(13,'  CHANGE TO: '),
     MSG06 DATA(16,XOFF,' **START TIME: '),
     MSG07 DATA(33,XOFF,' *AVERAGING INTERVAL (MINUTES): ');
```

```
MAG$TAPE: PROCEDURE(WORDVAL);
     DEC WORDVAL BYTE;
     OUT03 = NOT WORDVAL;
     OUT02 = 02H;   OUT02 = 00;
     CALL DELAY(0BBH);
   END MAG$TAPE;

TTY$BYTEOUT: PROCEDURE(OUTVAL);
     DEC (OUTVAL,I) BYTE, HALF(2) BYTE;
          HALF(0) = SHR(OUTVAL,4);
          HALF(1) = OUTVAL AND 0FH;
          DO I = 0 TO 1;
               IF HALF(I) > 9 THEN HALF(I) = HALF(I) + 37H;
               ELSE HALF(I) = HALF(I) + 30H;
               CALL TTYCHAR$OUT(HALF(I));
          END;
   END TTY$BYTEOUT;

PRINTOUT: PROCEDURE(INDEX);
     DEC (INDEX,TOP,BOT) BYTE, XPAD ADDRESS;
          TOP = SHL(((XPAD:=ANSWER(INDEX))/1000),4)
          + (XPAD MOD 1000) / 100;
          BOT = SHL(((XPAD MOD 100)/10),4) + (XPAD MOD 10);
          CALL TTY$BYTEOUT(TOP);
          CALL MAG$TAPE(TOP);
          CALL TTY$BYTEOUT(BOT);
          CALL MAG$TAPE(BOT);
   END PRINTOUT;

SPACES: PROCEDURE(NUMBER);
     DEC (I,NUMBER) BYTE;
          DO I = 1 TO NUMBER;
               CALL TTYCHAR$OUT(' ');
          END;
   END SPACES;
```

```
TTY$BITEIN: PROCEDURE BYTE;
     DEC (WORD,XPAD) BYTE;
          BI1:  WORD = 0;
          BI2:  IF (XPAD:= TYCHAR$IN) = 20H THEN
                DO;  CALL TTYCHAR$OUT(XPAD);
                     RETURN WORD;
                END;
          IF XPAD < 30H OR XPAD > 46H THEN
                DO;  CALL TTYCHAR$OUT(3FH);
                     GO TO BI1;
                END;
          CALL TTYCHAR$OUT(XPAD);
          IF XPAD < 40H THEN XPAD = (XPAD AND 0FH);
          ELSE XPAD = XPAD - 37H;
          WORD = SHL(WORD,4) OR XPAD;
          GO TO BI2;
   END TTY$BITEIN;

RESPONSE: PROCEDURE BYTE;
     DEC XPAD BYTE;
          R1:  CALL MESSAGE(.MSG01);
          IF (XPAD:=TYCHAR$IN) = 'Y' OR XPAD = 'N' THEN
                DO;  CALL TTYCHAR$OUT(XPAD);
                     IF XPAD = 'Y' THEN RETURN TRUE;
                     ELSE RETURN FALSE;
                END;
          CALL TTYCHAR$OUT(3FH);
          GO TO R1;
   END RESPONSE;
```

```
AVERAGE: PROCEDURE(INDEX) ADDRESS;
     DEC (INDEX,UPPER,LOWER,TOP,BOTTOM) BYTE, BOT ADDRESS;
          BOTTOM = (TOP:=SHL(INDEX,1)) + 1;
          UPPER = VALUE(TOP) / NSAMP;
          LOWER = (BOT:=VALUE(BOTTOM) +
          100* (VALUE(TOP) MOD NSAMP) ) / NSAMP;
          IF SHL(BOT MOD NSAMP,1) > NSAMP THEN
             LOWER = LOWER + 1;
          IF LOWER >= 100 THEN
               DO;  LOWER = LOWER - 100;
                    UPPER = UPPER + 1;
               END;
     RETURN(100*UPPER + LOWER);
   END AVERAGE;

ADC$PRINT: PROCEDURE(INDEX);
     DEC (K,K1,INDEX,UPPER,LOWER,IVAL) BYTE,
          (NUMB,BOTT) ADDRESS;
          K1 = (K:= SHL(INDEX,1)) +1;
          UPPER = VALUE(K) / NSAMP;
          LOWER = (BOTT:=VALUE(K1) + 100H *
               (VALUE(K) MOD NSAMP) ) / NSAMP;
          IF SHL(BOTT MOD NSAMP,1) > NSAMP THEN
               DO;  IF (LOWER:=LOWER+1) = 0 THEN
                    UPPER = UPPER + 1;
               END;
          NUMB = 125 * UPPER;
          LOWER = ROL(LOWER,1);
          IF CARRY THEN NUMB = NUMB + 63;
          LOWER = ROL(LOWER,1);
          IF CARRY THEN NUMB = NUMB + 31;
          NUMB = NUMB + SHR(LOWER,3);
          ANSWER(INDEX) = NUMB * MAX$ADC$VOLTS;
          CALL PRINTOUT(INDEX);
   END ADC$PRINT;
```

```
DUMPLINE: PROCEDURE;
     CALL TTY$CHAROUT(XOFF);
     CALL TTY$CHAROUT(' ');
   END DUMPLINE;

****************   MIDAS IIA MAIN PROGRAM   ***************

RESTART:
     CALL MESSAGE(.MSG00);
     IF RESPONSE THEN
          DO I = 0 TO LAST(INITLIST);
               SCANLIST(I) = INITLIST(I);
          END;

CHG$LOOP:
     CALL MESSAGE(.MSG02);
     IF RESPONSE THEN
          DO;  CALL MESSAGE(.MSG03);
               POSIT = BCD$TO$HEX(TTY$BITEIN) - 1;
               CALL MESSAGE(.MSG04);
               CALL TTY$BYTEOUT(SCANLIST(POSIT));
               CALL MESSAGE(.MSG05);
               SCANLIST(POSIT) = TTY$BITEIN;
               GO TO CHG$LOOP;
          END;

     CALL MESSAGE(.MSG06);
     START$TIME = 100H*TTY$BITEIN + TTY$BITEIN;
     CALL MESSAGE(.MSG07);
     INTERVAL = BCD$TO$HEX(TTY$BITEIN);
     NSAMP = SHL(INTERVAL,1);
     CALL DUMPLINE;
     CALL TTYCRLF;
     I = 0;
          DO WHILE SCANLIST(I) <> 0FFH;
               I = I + 1;
```

```
          END;
     ENDLIST = I - 1;

STRT$SAMPLE:
     IF CLOCK <> ZEROSECS THEN GO TO STRT$SAMPLE;
     IF HM <> START$TIME THEN GO TO STRT$SAMPLE;
   CLEAR: OUT00 = 40H;   OUT00 = 0;   OUT01 = 0;
     S1:  IF CLOCK THEN GO TO S1;
          DO I = 0 TO 79;
               VALUE(I) = 0;
          END;
          DO I = 0 TO 39;
               ANSWER(I) = 0;
          END;

     DO I = 1 TO NSAMP;
          S2:  IF NOT CLOCK THEN GO TO S2;
          K1 = 1;
          CALL ANEMOMETER;
          K = 8;
          DO M = 0 TO 15;
               OUT01 = SHL(M,4);
               CALL DELAY(77H);
               OUT02 = 08H;
               CALL DELAY(88H);
               CALL MULTI$SENSOR(ADC,80H,0,07H);
               K = K + 2;   K1 = K1 + 2;
               OUT02 = 00;
          END;
          L = 0;

          DO J = 0 TO ENDLIST;
               IF (LEVEL:=SHR(SCANLIST(J),4)) <> L THEN
                    DO;  OUT01 = (L:=LEVEL);
                         CALL LEVL$DELAY(99H);
                    END;
```

```
            DO CASE (SCANLIST(J) AND 07H);
                    VALUE(K1), VALUE(K) = DUMMY;
                    CALL MULTI$SENSOR(HUM,80H,0,0FH);
                    CALL MULTI$SENSOR(HTEMP,80H,0,0FH);
                    CALL QUARTZ;
                    CALL MULTI$SENSOR(BAROM,
                                        80H,80H,3FH);
            END;   * OF CASES  *
            K = K + 2;    K1 = K1 + 2;
      END;    * OF ENDLIST  *
      OUT01 = 0;
END;    * END OF SAMPLE  *

DO I = 0 TO 3;
      ANSWER(I) = AVERAGE(I);
END;
DO I = 20 TO 33;
      ANSWER(I) = AVERAGE(I);
END;
CALL TTY$CRLF;
CALL SPACES(2);
CALL MAG$TAPE(0FFH);
CALL TTY$BYTEOUT(HIGH(HM));
CALL TTY$BYTEOUT(LOW(HM));
CALL DUMPLINE;
CALL MAG$TAPE(HIGH(HM));
CALL MAG$TAPE(LOW(HM));
CALL TTY$CRLF;
CALL PRINTOUT(20);  CALL SPACES(4);  CALL PRINTOUT(21);
CALL DUMPLINE;
L = 0FFH;    K = 4;
DO I = 24 TO ENDLIST + 20 BY 3;
      CALL TTY$CRLF;
      DO J = 0 TO 2;
            CALL PRINTOUT(I-J);  CALL SPACES(2);
      END;
```

```
        CALL PRINTOUT(L:=L+1);    CALL SPACES(2);
             DO M = 0 TO 3;
                  CALL ADC$PRINT(K+M);    CALL SPACES(2);
             END;
   K = K + 4;
   CALL DUMPLINE;
   END;
   CALL TTY$CRLF;
   DO WHILE CLOCK <> ZEROSECS;
   END;
   GO TO CLEAR;
EOF
```

APPENDIX P

MIDAS IIA INTELLEC 8 COMPUTER PROGRAM FOR CASSETTE DATA TO PAPER TAPE

```
START  2000   JSU     46   PUNCH TIME VALUE
          1   NUMB    30   "
          2   PAGE    20   "
          3   JSU     46   PUNCH END OF LINE (EOL)
          4   EOL     40   CHARACTERS
          5   PAGE    20   "
          6   LAI     06   PUNCH LEVEL 0
          7   02      02   "
          8   JSU     46   "
          9   NCTR    50   "
          A   PAGE    20   "
          B   JSU     46   PUNCH END OF LINE (EOL)
          C   EOL     40   CHARACTERS
          D   PAGE    20   "
          E   LBI     0E   PUNCH LEVEL 1-4 LINES
          F   04      04   INCLUDING EOL'S
       2010   LAI     06   "
          1   08      08   "
          2   JSU     46   "
          3   LCTR    70   "
          4   PAGE    20   "
          5   JSU     46   GO TO START FOR
          6   START   00   ANOTHER SAMPLE
          7   PAGE    20   "
```

```
NUMB    2030    JSU     46    GET 2 DATA BYTES
           1    BYTE    90    AND PUNCH THEM
           2    PAGE    20    "
           3    JSU     46    "
           4    BYTE    90    "
           5    PAGE    20    "
           6    LBI     0E    PUNCH 2 SPACES AFTER
           7    20      20    THE 4 DIGITS OF DATA
           8    JSU     46    "
           9    PCH"B"  B5    "
           A    PAGE    3E    "
           B    JSU     46    "
           C    PCH"B"  B5    "
           D    PAGE    3E    "
           E    RET     07    "

EOL     2042    LBI     0E    PUNCH AN "XOFF" TO INDICATE
           1    13      13    AN END OF RECORD TO CP/CMS
           2    JSU     46    "
           3    PCH"B"  B5    "
           4    PAGE    3E    "
           5    LBI     0E    PUNCH A SPACE
           6    20      20    "
           7    JSU     46    "
           8    PCH"B"  B5    "
           9    PAGE    3E    "
           A    JSU     46    PUNCH A CR, LF.
           B    CRLF    95    "
           C    PAGE    3E    "
           D    RET     07    "

NCTR    2050    LHI     2E    POINT AT 2FF6(NUMB CTR)
           1    2F      2F    "
           2    LLI     36    "
           3    F6      F6    "
```

```
          4   LMA     F8   STORE # OF NUMBS
NC1       5   JSU     46   PUNCH A NUMB
          6   NUMB    30   "
          7   PAGE    20   "
          8   LHI     2E   POINT AT 2FF6
          9   2F      2F   "
          A   LLI     36   "
          B   F6      F6   "
          C   LBM     CF   PUT VALUE IN B
          D   DCB     09   AND DECREMENT
          E   RTZ     2B   RETURN IF = 0
          F   LMB     F9   STORE BACK IN NUMB CTR
       2060   JPU     44   GO PUNCH ANOTHER NUMB
          1   NC1     55   "
          2   PAGE    20   "

LCTR   2070   LHI     2E   POINT AT 2FF4 (LINES CTR)
          1   2F      2F   "
          2   LLI     36   "
          3   F4      F4   "
          4   NOOP    C0   NO OPERATION
          5   INL     30   STORE # OF NUMBS IN 2FF5
          6   LMA     F8   AND 2FF6
          7   INL     30   "
          8   LMA     F8   "
LC1       9   JSU     46   JUMP AND PUNCH ALL NUMBS
          A   NC1     55   ON ONE LINE
          B   PAGE    20   "
          C   JSU     46   PUNCH AN EOL
          D   EOL     40   "
          E   PAGE    20   "
          F   LHI     2E   POINT AT 2FF4 (LINES CTR)
       2080   2F      2F   "
          1   LLI     36   "
          2   F4      F4   "
```

```
          3   LBM     CF   GET LINES CTR,DECR.,
          4   DCB     09   AND RETURN IF =0
          5   RTZ     2B   "
          6   LMB     F9   ELSE STORE NEW VALUE
          7   INL     30   GET # OF NUMBS FOR NEXT LINE
          8   LAM     C7   "
          9   INL     30   "
          A   LMA     F8   STORE # OF NUMBS FOR NEXT LINE
          B   JPU     44   GO PUNCH ANOTHER LINE
          C   LC1     79   "
          D   PAGE    20   "

BYTE   2090   LHI     2E   POINT MEMORY AT 2FF0
          1   2F      2F   "
          2   LLI     36   "
          3   F0      F0   "
          4   LBM     CF   PUT DATA ADDRESS IN B AND C
          5   INL     30   "
          6   LCM     D7   "
          7   LHB     E9   POINT MEMORY AT DATA ADDRESS
          8   LLC     F2   "
          9   LDM     DF   PUT DATA IN D
          A   INL     30   INCREMENT LINE AND IF =00
          B   JSTZ    6A   THEN ALSO INCREMENT PAGE #
          C   INH     D0   "
          D   PAGE    20   "
          E   LBH     CD   SAVE INCR. ADDRESS IN B AND C
          F   LCL     D6   "
       20A0   LHI     2E   POINT MEMORY AT 2FF0
          1   2F      2F   "
          2   LLI     36   "
          3   F0      F0   "
          4   LMB     F9   STORE INCREMENTED DATA ADDRESS
          5   INL     30   "
          6   LMC     FA   "
```

```
            7   INL      30   PUT PAGE # OF STOP ADDRESS
            8   LAM      C7   IN A
            9   CPB      B9   COMPARE STOP PAGE # WITH
            A   JFZ      48   INCR. DATA PAGE # AND GO TO
            B   CONT     B5   CONT IF DATA PAGE LESS
            C   PAGE     20   "
            D   INL      30   PUT ENDING LINE # IN A
            E   LAM      C7   "
            F   ADI      04   COMPARE STOP LINE # WITH
         20B0   02       02   DATA LINE # AND GO TO
            1   CPC      BA   THE MONITOR IF EQUAL
            2   JTZ      68   "
            3   MONITOR  44   "
            4   PAGE     38   "
CONT        5   LAD      C3   TEST FOR DATA EQUAL FF.
            6   CPI      3C   IF SO GO TO BYTE
            7   FF       FF   "
            8   JTZ      68   "
            9   BYTE     90   "
            A   PAGE     20   "
            B   JSU      46   PUNCH DATA
            C   PCH"A"   7B   "
            D   PAGE     3E   "
            E   RET      07   "

INH      20D0   INH      28   INCREMENT "H"
            1   RET      07   "
```

LIST OF REFERENCES

1. Sturges, Lt. J.W., A MICROPROGRAMMABLE INTEGRATED DATA ACQUISITION SYSTEM (MIDAS), Aeronautical Engineer Thesis, Naval Postgraduate School, Monterey, California, 1975.

2. Biewer, M., THE DESIGNERS GUIDE TO PROGRAMMED LOGIC FOR MPS-800 SYSTEMS, PRO-LOG Corporation, 1974.

3. Pro-Log Corporation, MPS-300 SERIES DRAWINGS, by B. McCoy, 1973.

4. Pro-Log Corporation, WIRE LIST, MPS-805 CARD RACK, A100602 by B. McCoy, 1974.

5. Lee, E., THE PROM USERS GUIDE, Pro-Log Corporation, 1976.

6. Validyne Engineering Corporation, DB-99 DIGITAL BAROMETER, Bulletin DP 15-7/74, 1974.

7. Hewlett-Packard Corporation, INSTRUCTION MANUAL MODEL 2801A QUARTZ THERMOMETER.

8. American Instrument Company, HYGRODYNAMICS DIGITAL HYGROMETER INDICATORS, Instruction Manual H-210, 1973.

9. Memodyne Corporation, INSTRUCTION MANUAL: MODEL 173.

10. Teletype Corporation, TECHNICAL MANUAL FOR 32 AND 33 TELETYPEWRITER SETS, Bulletin 273B, Vol. 1.

11. Datel Systems Inc., 16 CHANNEL ANALOG MULTIPLEXER: MM16, Bulletin X160515310.

12. Datel Systems Inc., SAMPLE AND HOLD MODEL SHM-4, Bulletin 3447310K.

13. Datel Systems Inc., ANALOG-TO-DIGITAL CONVERTER MODEL ADC-E SERIES, Bulletin ADECT15404, 1974.

14. Intel Corporation, 8008 and 8080 PL/M PROGRAMMING MANUAL, Revision A, 1975.

15. Intel Corporation, 3080 PL/M COMPILER OPERATORS MANUAL, Revision A, 1975.

16. Intel Corporation, INTELLEC 8/MOD8 MICROCOMPUTER SYSTEM OPERATOR'S MANUAL VERSION II.

17. Naval Postgraduate School (W.R. Church Computer Center), Technical Note No. 0211-24, CP/CMS USERS GUIDE AT THE NAVAL POSTGRADUATE SCHOOL, by K.B. Strutynski, October, 1973.

18. Datamedia Corporation, ELITE 2500 INSTRUCTION MANUAL.

19. Addmaster Corporation, MODEL 606/608 STAND ALONE READER.

20. Pro-Log Corporation, SERIES 81 PORTABLE PROM PROGRAMMER'S MANUAL, 1974.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Chairman, Department of Aeronautics Naval Postgraduate School Monterey, California 93940	1
4.	Chairman, Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93940	1
5.	Dr. Thomas M. Houlihan, Code 69Hm Associate Professor Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93940	5
6.	Mr. T. Christian, Code 69 Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93940	1
7.	Mr. R. Garcia, Code 012A Department of Meteorology Naval Postgraduate School Monterey, California 93940	1

8. LT. J.W. Sturges, USN 1
Helicopter Combat Support Squadron 3
HC-3 DET. 112
FPO San Francisco, 96601

9. LCDR. J.R. Plunkett, USN 1
Carrier Airborne Early Warning
Training Squadron 110
FPO San Francisco, 96601